

Python para Pentesters

Daniel Echeverri Montoya

Más de 1000 ejemplares vendidos



0xWORD
www.0xWORD.com

Índice

Introducción

En los últimos años, el término "*hacker*" ha sido bastante controvertido y se ha asociado a personas que utilizan herramientas y medios informáticos para realizar actividades delictivas. Esto ha dado lugar a un temor generalizado sobre dicho colectivo y ha sido una situación que desafortunadamente, se ha ido agravando con la ayuda de los medios de comunicación convencionales los cuales no suelen distinguir entre un delincuente y un *hacker*.

Seguramente los lectores de este libro entienden perfectamente la diferencia, sin embargo, hay muchas personas que opinan que un delincuente y un *hacker* son exactamente lo mismo. Esto es algo lamentable, pero aún existen comunidades de usuarios que intentan conservar los principios y la filosofía *hacker* con sus mejores herramientas: La información y el conocimiento. Dichas comunidades se encargan de publicar documentación sobre temas relacionados con la seguridad informática y desarrollan herramientas que pueden ser utilizadas para apoyar los procesos de auditoria ejecutados por *pentesters* profesionales. El enfoque de estas comunidades, es proveer el conocimiento y los medios necesarios para que un *pentester* pueda detectar cualquier tipo de fallo antes que un atacante.

Uno de los principales objetivos de este libro, es exponer algunas de las técnicas utilizadas por los atacantes en *Internet* para comprometer sistemas y posteriormente tener control total sobre dichas máquinas. ¿Por qué motivo? Porque para un *pentester* profesional, entender cómo piensa y actúa un atacante, es muy importante para evitar o detectar intrusiones. Podría decirse, que la mejor forma de aprender a defender efectivamente una aplicación o un servicio, es conociendo el modo de actuar de un adversario, aprendiendo sus técnicas y herramientas de uso común para poder intentar anticiparse y tomar las medidas preventivas oportunas. Esto se conoce como seguridad ofensiva y es un enfoque muy potente, ya que le permite a un *pentester* tener una visión global sobre las técnicas que pueden ser utilizadas para comprometer un sistema y las medidas defensivas que deben adoptarse.

Por otro lado, muchas de las herramientas y librerías cuyo enfoque se centra en la seguridad informática, se encuentran escritas en lenguaje *Python* y la principal razón de esto, es debido a que *Python* ha sido un lenguaje utilizado mayoritariamente por *hackers*, los cuales prefieren lenguajes de programación sencillos y potentes; *Python* cumple con ambos requisitos.

Algunas de las herramientas y librerías más destacadas hoy en día y que se encuentran escritas en lenguaje *Python* se listan a continuación:

- *Scapy.*
- *Sulley Framework.*



- *PyDBG.*
- *Cuckoo Sandbox.*
- *Volatility Framework.*
- *W3AF.*
- *Paramiko.*
- *Mechanize.*
- *BeautifulSoup.*
- *Python-nmap.*
- *Stem.*
- *Twisted.*
- *Pamei.*
- *Veil Framework.*
- *SET (Social Engineering Toolkit).*

En el listado anterior, solamente se incluyen algunas de las librerías y herramientas más conocidas en lenguaje *Python*, sin embargo existen muchas más que no se han listado y que han tenido un impacto muy positivo en la forma en la que se crean y auditan algunas de las aplicaciones y protocolos de uso común.

La potencia y simplicidad de *Python*, le convierte en un lenguaje de programación ideal para crear pruebas de concepto rápidas que permitan estructurar una idea y posteriormente demostrarla. Además, ha sido un lenguaje del que se ha hablado muchísimo y que han recomendado *hackers* y personas con mucho renombre en el campo de la seguridad informática desde hace varios años, un ejemplo claro se encuentra en el documento de *Eric S. Raymond* titulado "*How to become a Hacker*" (Como convertirse en un *Hacker*) en el que habla sobre la filosofía, la cultura y los principios psicológicos más destacados en los *Hackers*. En uno de los apartados de dicho documento, habla sobre las habilidades que desarrolla un *hacker* en su proceso de aprendizaje continuo y menciona lenguajes como *Python* y *C* para crear programas con fines centrados en la seguridad informática y el *Hacking*. Dicho documento, se encuentra disponible en la siguiente ruta:
http://www.catb.org/~esr/faqs/hacker-howto.html y se recomienda su lectura.

Aunque muchas de las herramientas y librerías desarrolladas en *Python* enfocadas a la seguridad informática, tienen como objetivo apoyar los procesos de *pentesting* para encontrar y corregir vulnerabilidades, evidentemente, también han sido utilizadas por atacantes para detectarlas y explotarlas, por este motivo es importante contar con los conocimientos necesarios para anticiparse a las acciones potencialmente peligrosas de los atacantes en *Internet*. Un buen *pentester* y en general, cualquier persona que se dedique profesionalmente a la informática, preferiblemente debería saber programar para crear sus propias utilidades o ayudar a corregir fallos en herramientas existentes.

Por otro lado, es importante resaltar que la intención de este libro es ayudar al lector a afinar sus habilidades en programación con *Python* y proveer los conocimientos necesarios para entender



las principales técnicas utilizadas por delincuentes en *Internet*. No se trata de un manual para cometer delitos informáticos, sino todo lo contrario, para intentar prevenirlos y alertar sobre sus consecuencias.

La mejor forma de aprender a proteger cualquier recurso informático, consiste precisamente en aprender a pensar como lo hace un atacante para conseguir anticiparse. Tal como decía *Sun Tzu* en "El arte de la guerra": "Si conoces a tu enemigo y te conoces a ti mismo, ni en cien batallas correrás peligro; si no conoces a tu enemigo, pero te conoces a ti mismo, perderás una batalla y ganarás otra; si no conoces a tu enemigo ni te conoces a ti mismo, correrás peligro en cada batalla".

Tienes en tus manos un documento que ha sido el resultado de varias horas de esfuerzo y dedicación, el cual espero que utilices de forma responsable en un entorno controlado y con la autorización correspondiente según sea el caso. No es mi intención proporcionar una guía para delincuentes, sino un documento para profesionales y entusiastas de la seguridad informática que les guste programar y afinar sus conocimientos, una guía para personas que les gusta saber cómo funcionan las cosas y que se encuentran en un proceso continuo de aprendizaje. Es un libro que espero que sea de tu agrado, que te aporte conocimientos y te ayude a desarrollar herramientas que luego puedas compartir con toda la comunidad *hacker*.

Saludos y *Happy Hack*!!

Adastra. (@jdaanial)



Capítulo I
Recolección de Información y Escaneo

1.1 La información es la base en el mundo del hacking

En la era digital y con el inmenso auge de herramientas, tecnologías y avances, no cabe duda que una de las mejores armas que tienen los profesionales de la seguridad informática es precisamente la información y el sentido común. Sin embargo, uno de los retos más complejos a los que nos enfrentamos hoy en día, es determinar qué información es relevante y cuál representa simplemente datos sin trascendencia. El caso de los administradores de sistemas, es un claro ejemplo de esta situación, se trata de profesionales que tienen que enfrentarse diariamente a enormes ficheros de *logs*, alarmas de sistemas de registro y detección de intrusiones, entre muchas otras cosas. Para ellos su pericia, capacidad de análisis y criterio, son los elementos que realmente puede hacer la diferencia entre un sistema seguro y uno comprometido. Por otro lado, un atacante se enfrenta exactamente al mismo reto, su capacidad de extraer y procesar información sobre el sistema objetivo, es vital para poder encontrar cualquier tipo de fallo o fuga de información sensible, su capacidad de recopilar y utilizar la información que obtiene sobre su objetivo es lo que le permite alcanzar sus propósitos.

Además de lo anterior, cuando se habla de la anatomía de un ataque y de seguridad ofensiva, la recopilación y análisis de información es un proceso transversal que se debe implementar en todas las etapas, desde el reconocimiento inicial del objetivo hasta las etapas finales que consisten principalmente, en limpiar rastros y crear puertas traseras para intentar garantizar accesos futuros.

Dicho esto y sin perder de vista el contexto de este libro, en *Python* existen una gran variedad de herramientas y librerías que permiten extraer información para realizar actividades de reconocimiento. Los siguientes apartados detallarán algunas de las más comunes y utilizadas para la recolección y categorización de información.

1.2 Definición del sistema objetivo

Desde el punto de vista de un atacante, una vez se ha definido cuál será el sistema objetivo, lo primero es intentar delimitar los puntos principales de acceso y la información que se encuentra disponible de forma pública. En esta etapa inicial, puede resultar útil para un atacante, cualquier tipo



de información que le permita entender las actividades del objetivo. Por ejemplo, en el caso de que sea una organización, es vital recopilar información relacionada con sus empleados, ubicación de oficinas, las actividades a las que se dedica, organigramas, clientes, etcétera. Todos estos detalles son muy importantes para un atacante a la hora de planificar sus actividades y dada la naturaleza pública de dicha información, puede ser utilizada por cualquiera.

Ahora bien, desde el punto de vista de un *pentester* profesional que ha sido contratado para auditar las medidas de seguridad de una organización, es muy importante delimitar adecuadamente el alcance de sus actividades y obtener por escrito, una autorización firmada indicando concretamente las actividades que se pueden llevar a cabo. Ese tipo de documentos son muy importantes ya que le permiten a un *pentester* realizar ataques controlados en un contexto legal y legítimo.

1.3 Recolección de Información con Python

Automatizar el proceso de recolección de información puede conseguirse utilizando las librerías incluidas directamente en el lenguaje, así como algunas de las disponibles por terceros. A continuación se detalla el uso de algunas de las más útiles para cumplir con este objetivo.

1.3.1 Extraer información en servidores DNS utilizando DNSPython

Con esta librería es posible realizar consultas a servidores *DNS* del mismo modo que es posible hacerlo con herramientas como *dig*, *fierce* o *nslookup*, con la ventaja de que el resultado de las consultas se puede controlar desde *Python* y posteriormente dicha información puede ser utilizada para distintos fines directamente desde un *script*. El sitio *web* oficial de esta librería es el siguiente: *http://www.dnspython.org/*.

Para instalarla, solamente es necesario ejecutar el *script setup.py* con el argumento *install*.

```
# Python setup.py install
```

Ahora, se pueden realizar diferentes tipos de consultas que permitirán obtener información sobre un dominio concreto.

```
#python
Python 2.6.6 (r266:84292, Dec 26 2010, 22:31:48)
[GCC 4.4.5] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import dns
>>> import dns.resolver
>>>ansA,ansMX,ansNS,ansAAAA=(dns.resolver.query('google.com','A'),dns.resolver.
query('google.com','MX'),dns.resolver.query('google.com', 'NS'), dns.resolver.
query('google.com', 'AAAA'))
>>> print ansA.response.to_text()
id 53047
opcode QUERY
```

```
rcode NOERROR
flags QR RD RA
;QUESTION
google.com. IN A
;ANSWER
google.com. 63 IN A 173.194.34.200
google.com. 63 IN A 173.194.34.201
google.com. 63 IN A 173.194.34.199
google.com. 63 IN A 173.194.34.196
google.com. 63 IN A
google.com. 63 IN A 173.194.34.192
google.com. 63 IN A 173.194.34.197
google.com. 63 IN A 173.194.34.193
google.com. 63 IN A 173.194.34.195
google.com. 63 IN A 173.194.34.199
google.com. 63 IN A 173.194.34.206
;AUTHORITY
;ADDITIONAL
>>> print ansMX.response.to_text()
id 46859
opcode QUERY
rcode NOERROR
flags QR RD RA
;QUESTION
google.com. IN MX
;ANSWER
google.com. 99 IN MX 10 aspmx.l.google
google.com. 99 IN MX 20 alt1.aspmx.l.google.com
google.com. 99 IN MX 50 alt4.aspmx.l.google.com
google.com. 99 IN MX 40
google.com. 99 IN MX 30
;AUTHORITY
;ADDITIONAL
alt1.aspmx.l.google.com. 50 IN A 173.194.70.26
alt1.aspmx.l.google.com. IN AAAA
alt4.aspmx.l.google.com. 105 IN A 74.125.25.26
aspmx.l.google.com. 102 IN A 74.125.143.26
alt2.aspmx.l.google.com. 180 IN A 173.194.69.26
aspmx.l.google.com. IN AAAA
>>> print ansNS.response.to_text()
id
opcode QUERY
rcode NOERROR
flags QR RD RA
;QUESTION
google.com. IN NS
;ANSWER
google.com. 194408 IN NS ns2.google.com
google.com. 194408 IN NS ns3.google.com
google.com. 194408 IN NS ns4.google.com
google.com. 194408 IN NS ns1.google.com
;AUTHORITY
```

```
;ADDITIONAL
ns3.google.com. 192065 IN A [illegible]
ns4.google.com. 192066 IN A [illegible]
ns1.google.com. 191898 IN A [illegible]
ns2.google.com. 191940 IN A [illegible]
>>> print ansAAAA.response.to_text()
[illegible] 38970
opcode QUERY
rcode NOERROR
flags QR RD RA
;QUESTION
google.com. IN AAAA
;ANSWER
google.com. 232 IN AAAA [illegible]
;AUTHORITY
;ADDITIONAL
```

En este caso se ha utilizado *DNSPython* para ejecutar consultas sobre varios tipos de registros *DNS*, en concreto sobre los registros *IPV4 (A), IPV6 (AAAA), Nameservers (NS) y MailServers (MX)*

Además de extraer este tipo de información, otra prueba bastante común contra servidores *DNS*, es intentar realizar una transferencia de zona, que consiste en permitir a un servidor *DNS* secundario actualizar su base de datos de zona a partir de un servidor *DNS* primario. Se trata de una característica que permite a un servidor *DNS* secundario (o espejo) garantizar la disponibilidad del servicio cuando el servidor *DNS* principal no puede atender más peticiones o se encuentra caído. Cuando un servidor *DNS* no se encuentra correctamente configurado puede proveer una copia completa de sus registros a cualquiera que lo solicite, permitiendo de esta forma, la divulgación de información sensible que no debería ser expuesta de forma pública.

```
>>> import dns.query
>>> import dns.zone
>>> [illegible] = dns.zone.from_xfr(dns.query.xfr('[illegible]', '[illegible]hackingway.com'))
```

En este caso, se intenta realizar una transferencia de zona entre el servidor *DNS* principal y un dominio controlado por el atacante. En el caso de que el servidor no se encuentre correctamente configurado, la transferencia de zona se realizará correctamente y en el dominio del atacante se almacenará una copia de los registros del servidor *DNS* principal.

1.3.2 Ejecutar consultas WHOIS con pythonwhois

Aunque para los más neófitos *Internet* puede parecer una red carente control y completamente descentralizada, la realidad es que existen organizaciones que se encargan de gestionar la interoperabilidad de los sistemas que funcionan en la red e intentan prevenir los conflictos que puedan producirse con las direcciones *IP*, así como también la gestión de los nombres de dominios, parámetros de protocolos como *DNS* y números de puertos. Este tipo de funciones son llevadas a cabo desde hace varios años por una organización sin ánimo de lucro compuesta por empresas, universidades y comunidades de usuarios llamada *ICANN (Internet Corporation for Assigned Names and Numbers)*. Aunque antiguamente estos detalles técnicos eran gestionados completamente por

el gobierno de los Estados Unidos de América con la *IANA (Internet Assigned Numbers Authority)* en la actualidad dichas actividades son ahora supervisadas y coordinadas por la *ICANN*. De hecho, IANA es ahora un departamento complementario de la *ICANN*.

Aunque los servidores centrales de la *ICANN* realizan actividades de gestión sobre dominios, direcciones *IP*, servidores, entre otras cosas, la información necesaria para resolver direcciones y dominios se encuentra diseminada por *Internet* en múltiples servidores *WHOIS*.

WHOIS es un protocolo que permite realizar consultas con el fin de obtener información detallada sobre el propietario de un dominio, fechas de creación y caducidad, teléfonos de contacto e incluso direcciones y códigos postales. Desde luego es una muy buena fuente de información para un atacante y un punto de inicio adecuado para perfilar un objetivo. Realizar consultas *whois* es un procedimiento bastante trivial en sistemas basados en Unix ejecutando utilidades por la línea de comandos, sin embargo existen una gran variedad de servicios en *Internet* que permiten consultar los registros *whois* para un dominio determinado, por ejemplo: http://whois.net/

No obstante, para un atacante es conveniente tener toda esta información en estructuras de datos que puedan ser utilizadas posteriormente desde un *script*. Es aquí donde es posible utilizar la librería *pythonwhois* que permite realizar consultas *whois* contra un dominio concreto y posteriormente almacenar los resultados en una estructura de datos consistente y fácil de utilizar. Esta librería puede ser descargada desde el *packet index* de *Python* en la siguiente dirección: *https://pypi.python.org/pypi/pythonwhois*.

El procedimiento de instalación de esta librería, como muchas otras disponibles en *Python*, consiste simplemente en ejecutar el *script setup.py* con el argumento *install*.

```
# python setup.py install
```

Una vez instalada la librería, es posible importar el módulo principal que permite emplear las clases y utilidades necesarias para realizar consultas.

```
#python
Python 2.6.6 (r266:84292, Dec 26 [illegible])
[GCC 4.4.5] on linux2
Type "help", "copyright", "credits" [illegible] for more information
>>> import pythonwhois
```

Ahora, se puede utilizar la *API* de *pythonwhois* para extraer información desde cualquier *script* en *Python*.

```
>>> pythonwhois.net.get_root_server('google.com')
'[illegible]sign-grs.com'
>>> whois = pythonwh[illegible].get_whois('google[illegible]')
>>> whois.keys()
['status', 'updated_date', [illegible], 'expiration_date', 'crea
tion_date', 'raw', 'whois_server', 'registrar', 'emails']
>>> whois.values()
[['clientUpdateProhibited', [illegible]
ted'], [datetime.datet[illegible]
[illegible] View', 'fax': '+1.6506188571', [illegible], 'phone'
```

```
'+1.6506234000', 'street' [illegible] '1600 Amphitheatre Parkway', 'country' [illegible] 'US', 'pos[illegible]
code': '94043', 'organization' [illegible] 'Google [illegible]', [illegible] 'dns-admin@google.com'
'tech': {'city': 'Mountain View', 'fax' [illegible], [illegible] 'DNS Admin'
[illegible]
```

Con la función *pythonwhois.net.get_root_server()* es posible recuperar el servidor raiz para [illegible] dominio determinado y con *pythonwhois.get_whois()* un diccionario con toda la información s[illegible] el dominio.

El el caso de que sea necesario obtener la respuesta cruda, es posible ejecutar la funcion *pythonw[illegible] get_whois_raw()* la cual retorna toda la información sobre un dominio especifico en una lista [illegible] respuestas, donde cada elemento de la lista, corresponde a cada uno de los servidores *WH[illegible]* consultados.

```
>>> import pp[illegible]
>>> wh[illegible] get_whois_raw("google.com")
>>> pr[illegible]
[illegible] \nRegistrar WHOIS Server: whois[illegible]
[illegible] com\nUpdated Date: [illegible]
[illegible] \nRegistrar Registrat[illegible]
[illegible] \nRegistrar: MarkMonitor, Inc.\nReg[illegible]
IANA ID: 292\nRegistrar
[illegible]
```

La información que se puede recolectar de los servidores *WHOIS* es muy variada y en todos [illegible] casos, sumamente informativa para un atacante

1.3.3 GeoLocalizacion del objetivo con PyGeoIP

Los *SIG* (Sistema de Información Geografico) han sido utilizados desde hace algunos años [illegible] localizar un punto determinado en un espacio geografico. Esta tecnologia también ha perm[illegible] relacionar direcciones *IP* públicas con redes de datos y localizaciones concretas en todo el mun[illegible]

En la actualidad existen bases de datos disponibles de forma publica, que asocian millones [illegible] direcciones *IP* de servidores en *Internet* con el pais y la ciudad donde se encuentran. Para un atac[illegible] el uso de dichas bases de datos supone una forma de validar la informacion que ha podido recole[illegible] con las consultas *WHOIS* realizadas previamente o de complementar el conocimiento ya dispon[illegible] sobre el objetivo.

La libreria *PyGeoIP* es en realidad un envoltorio de la libreria *geo-ip-c* que permite enco[illegible] información geográfica a partir de una direccion *IP* determinada. Dicha libreria depende directame[illegible] de las bases de datos *GeoLite* las cuales son actualizadas frecuentemente y pueden ser descarga[illegible] libremente desde su sitio *web* oficial *http://dev.maxmind.com/geoip/legacy/geolite/*

PyGeoIp se puede descargar desde su repositorio *GIT* en *https://github.com/appliedsec/pygeoip*

Del mismo modo que se ha indicado anteriormente para otras librerías, si se desea instalar *pygeoip* solamente es necesario ejecutar el *script setup.py* con el argumento *install*.

Una vez instalada, es necesario descargar las bases de datos *GeoLite* que pueden ser *Country*, *Country IPv6*, *City*, *City IPv6*, *ASN* y *ASN IPv6*.

```
#python
Python 2.6.6 (r266:84292, Dec 26 2010, 22:31:48)
[GCC 4.4.5] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import pygeoip
>>> import pprint
>>> gi = pygeoip.GeoIP('GeoLiteCity.dat')
>>> gi.country_code_by_name('google.com')
'US'
>>> pprint.pprint(gi.record_by_addr('173.194.34.192'))
{'area_code': 650,
 'city': u'Mountain View',
 'continent': 'NA',
 'country_code': 'US',
 [illegible]
 'country_name': [illegible],
 'dma_code': 807,
 'latitude': 37.4[illegible],
 [illegible]
 'metro_code': [illegible]co, CA',
 'postal_code': [illegible],
 'region_code': u'CA',
 'time_zone': 'America/Los_Angeles'}
>>> gi.country_name_by_addr('173.194.34.192')
'United States'
>>> gi.time_zone_by_addr('173.194.34.192')
'America/Los_Angeles'
```

Como se puede apreciar, utilizando la base de datos *GeoLiteCity.dat* es posible extraer información muy valiosa si tenemos la dirección *IP* o el nombre de dominio del objetivo.

Las otras bases de datos disponibles también deben ser consideradas, ya que contienen información valiosa que puede ser muy útil para un atacante.

```
>>> giv6 = pygeoip.GeoIP('GeoIPv6.dat')
>>> giv6.country_code_by_addr([illegible]1006')
'IE'
>>> giv6.country_name_by_addr('2a00:1450:400[illegible]:c01')
'Ireland'
>>> giAsn = pygeoip.GeoIP([illegible].dat')
>>> giAsn.isp_by_name('google.com')
u'AS15169 Google Inc.'
```

Alternativamente, también existen múltiples servicios online que permiten obtener este tipo de información como es el caso de *http://myip.es/* o *http://www.melissadata.com/lookups/iplocation.asp*

En dichos servicios es posible obtener información bastante fiable sobre la localización geográfica de un servidor en función a su dirección *IP*, no obstante, dicha información se enseña directamente en el navegador y para un atacante puede resultar interesante obtener dicha información directamente en sus *scripts* en *Python*. Para ello, puede utilizar algunas de las librerías disponibles para extraer información desde páginas *web* tales como *Mechanize* y *BeautifulSoup*.

1.4 Utilizar motores de búsqueda para recolectar información sobre un objetivo

Los motores de búsqueda como *Google*, *Yahoo!* o *Bing*, suelen indexar muchísima información sobre organizaciones y dominios. Dicha información en muchos casos no debería ser expuesta de forma pública y tales fugas pueden beneficiar enormemente a un atacante a la hora de perfilar víctimas potenciales.

1.4.1 Google Hacking

Además de ser uno de los motores de búsqueda más utilizado en todo el mundo, es también una herramienta muy útil para encontrar información concreta sobre sitios *web* en *Internet*. Una de las sus principales características, es que cuenta con una serie de operadores que permiten hacer búsquedas muy específicas. Lo que hoy en día se conoce como *Google Hacking* fue inicialmente introducido por *Johnny Long*, quien ha sido uno de los principales investigadores en el campo de recolección de información utilizando *Google*. Sin embargo, su evolución ha sido bastante interesante, hasta el punto que el proceso de búsqueda y recolección se encuentra ahora automatizado por herramientas y bases de datos con cadenas de búsqueda que permiten localizar objetivos vulnerables rápidamente.

Es posible que en un sitio *web* determinado existan fugas de información sensible y esa información puede ser capturada y almacenada por *Google*, algo que hoy en día es bastante conocido y aprovechado por usuarios maliciosos en *Internet*. Para controlar los resultados que pueden arrojar las búsquedas de *Google*, es necesario conocer el funcionamiento de algunos de los filtros disponibles a la hora de realizar consultas. A continuación se listan algunas de las palabras reservadas que permiten establecer dichos filtros.

Operadores en Google

Operador	Descripción	Uso	Resultado
site	Permite indicar el sitio *web* sobre el cual se realizará la búsqueda.	*site:securityfocus.com exploit XP SP2*	Búsqueda de la cadena "*exploit XP SP2*" en el sitio "*securityfocus.com*"

Operador	Descripción	Uso	Resultado
intitle	Busca un texto determinado en el título de la página	*intitle: FreePBX 2.10.0*	Búsqueda de páginas *web* que contengan en su título el texto "*FreePBX 2.10.0*"
inurl	Con este operador se pueden filtrar los resultados de una búsqueda, enseñando solamente aquellas coincidencias que contengan en la *URL* el texto indicado en el operador	*site: securityfocus.com inurl: bid kernel 2.6*	Búsqueda de páginas *web* que contengan en su *URL* el texto "*bid*" filtrando por las palabras "*kernel 2.6*" y para el sitio *web securityfocus.com*
ext	Permite buscar documentos con una extensión determinada	*site: www.target.com ext: pdf*	Búsqueda de cualquier documento *PDF* en el dominio *www.target.com*

Estos son solo algunos de los operadores disponibles para hacer búsquedas en *Google*, sin embargo, como se ha comentado en líneas anteriores, actualmente existe una base de datos bastante completa de *Google Dorks* llamada *Google Hacking Database*.

HDB es un repositorio bastante completo de *google dorks* y puede encontrarse en el siguiente enlace: *http://www.exploit-db.com/google-dorks*. Una de las ventajas de esta base de datos, es que todos los *dorks* se encuentran ordenados por categorías, lo que permite filtrar fácilmente aquellos que puedan resultar interesantes para un objetivo determinado. Las categorías que comprende la base de datos son:

- *Footholds*
- Directorios sensitivos.
- Ficheros vulnerables.
- Mensajes de error.
- Ficheros con *passwords*.
- Páginas de *login* a portales.
- Ficheros con nombres de usuarios.
- Detección de Servidor *web*.
- Servidores vulnerables.
- Ficheros con información interesante.
- Información sensitiva sobre sitios de compra online.
- Información de dispositivos de red.

...mo se puede apreciar, existe una gran cantidad de filtros en esta base de datos de *dorks*, pero ...no es todo, también existe una herramienta para la generación de dorks contra un objetivo ...minado y de esta forma se pueden ejecutar ataques dirigidos contra un objetivo concreto, ...na herramienta es *googleDB-tool* y se encuentra disponible en la siguiente dirección: *http://...secpoint.com/freetools/*. La última versión a la fecha de escribir este documento es la 1.5 y

permite generar listas de cadenas de búsqueda para *google* que pueden ser utilizadas para encontrar información interesante o incluso vulnerabilidades en el sistema objetivo. Cuenta con una pequeña base de datos interna que consiste simplemente de un listado de ficheros con *google dorks* y todos estos ficheros se encuentran almacenados en el directorio "*db*" ubicado en el directorio raíz de la herramienta. En dicho directorio se pueden incluir nuevos ficheros con dorks que correspondan a una categoría distinta a las existentes. Si se une esta herramienta con los *dorks* existentes en *Google Hacking Database*, el resultado puede ser muy interesante a la hora de realizar ataques dirigidos.

Después de descargar y descomprimir el paquete donde se encuentra la herramienta, es posible ver las opciones que soporta y algunos ejemplos de su uso.

```
python googleDB-tool.py -h
Usage: googleDB-tool.py <source file> [-s site [illegible]

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -o [illegible]
                        [illegible]
  -s SITE, --[illegible] generate queries for the SITE
  -m [illegible]
                        generate queries for multiple sites [illegible]
  -q, --query           generate google query urls for each
  -t, --html            generate output in HTML format [illegible]

SecPoint.com Google Penetration Testing Hack Database v. 1.5
```

El siguiente ejemplo, enseña el uso de la herramienta y genera un fichero de texto llamado "*salida.txt*" con algunos *google dorks* para encontrar páginas de *login* en el dominio "*www.objetivo.com*". Por otro lado, el fichero "*login_pages.txt*" se encuentra en la base de datos que utiliza la herramienta bajo el directorio "*db*" mencionado en párrafos anteriores.

```
python googleDB-tool.py [illegible] pages.txt" -o salida.txt -s www.objetivo.com
```

El siguiente ejemplo es bastante similar al anterior, con la diferencia que permite generar *google dorks* en base al fichero "*network_or_vulnerability_data.txt*" el cual contiene cadenas para buscar dispositivos y elementos de red vulnerables en el objetivo especificado.

```
python googleDB-tool.py "network_or_vulnerability_data.txt" -o salida.txt -s www.
objetivo.com
```

También es posible utilizar un fichero de texto para especificar múltiples objetivos.

```
python googleDB-tool.py "network_or_vulnerability_data.txt" -o salida.txt -m obje-
tivos.txt
```

1.4.2 Shodan Hacking

Shodan (*http://www.Shodanhq.com*) es conocido como "*el Google de los hackers*" y no es para menos, ya que se trata de un potente motor de búsquedas que permite encontrar servidores y dispositivos en *Internet* que ejecutan servicios muy concretos. A diferencia de los buscadores convencionales,

Shodan se encarga de indexar en su base de datos interna las cabeceras y *banners* correspondientes a servidores que se encuentran en ejecución en *Internet* y que han sido indexados en la base de datos de *Shodan*. Del mismo modo que navegadores como *Google*, *Shodan* cuenta con una serie de filtros que permiten restringir los resultados de las consultas y algunos de los más populares, se listan en la siguiente tabla.

Filtros en Shodan

Filtro	Descripción
city	En los resultados de la búsqueda, solamente aparecerán aquellos que corresponden con la ciudad indicada.
country	En los resultados de la búsqueda, solamente aparecerán aquellos que corresponden con el país indicado.
geo	Permite encontrar los dispositivos que se encuentran en el radio definido por latitud y longitud especificada.
hostname	En los resultados de la búsqueda, solamente aparecerán aquellos que corresponden con el nombre de dominio indicado.
net	Permite realizar búsquedas dirigidas a segmentos de red concretos.
os	Permite filtrar los resultados con un sistema operativo determinado.
port	Permite filtrar los resultados con un puerto determinado.

Se trata de una lista no exhaustiva y en la medida de que *Shodan* va creciendo, el número de filtros puede ser mayor, con lo cual, se anima al lector a leer la documentación oficial donde se incluyen los filtros soportados a la fecha *www.Shodanhq.com/help/filters*. Por otro lado, una característica que convierte a *Shodan* en una herramienta imprescindible para un atacante o un *pentester*, es que cuenta con una *API* que permite que desarrolladores en lenguajes de programación como *Ruby*, *Perl* o *Python*, puedan utilizar *Shodan* de forma programática, algo que desde luego resulta sumamente útil.

Para utilizar la *API*, es necesario tener una cuenta de usuario válida y de esta forma obtener una *Developer Key*. El uso de las características básicas de *Shodan* no supone ningún coste para un desarrollador, sin embargo existen *add-ons* que incluyen características muy interesantes, como por ejemplo la capacidad de realizar búsquedas específicas para servicios como *Telnet* o *HTTPS*, acceso a todos los filtros disponibles desde la *API* y acceso sin restricciones a los resultados de las consultas. Se trata de características que no son nada despreciables y que pueden ser adquiridas por un costo muy bajo.

Descargar e instalar la librería de *Shodan* para *Python* sigue el mismo patrón que muchas de las librerías disponibles para este lenguaje. Es posible hacerlo utilizando *easy_install* o directamente ejecutando el *script setup.py* con el argumento *install*. Instrucciones más detalladas sobre el proceso de instalación se pueden encontrar en la documentación oficial: *https://Shodan.readthedocs.org/en/latest/tutorial.html#installation*.

El uso más básico de la *API* de *Shodan* para *Python* consiste en crear una instancia de la clase *Shodan.Shodan* especificando como único argumento una *Developer Key* válida que se encuentra asociada a una cuenta de usuario.

Script: ShodanSimpleSearch.py

```
#!/usr/bin/env python
import Shodan
def ShodanTest():
    try:
        ShodanKeyString = "API KEY"
        ShodanApi = Shodan.Shodan(ShodanKeyString)
        results = ShodanApi.search("apache")
        for result in results['matches']:
            print 'IP: %s' % result['ip_str']
            print result['data']
            print ''
    except Shodan.APIError, e:
        [illegible]
if __name__ == "__main__":
    ShodanTest()
```

Con el *script* anterior se utiliza la *API* de *Shodan* para buscar por la palabra clave *"apache"* y posteriormente se pintan todos los resultados por pantalla. En este caso, se utiliza la función *search* que permite realizar cualquier tipo de búsqueda, incluso empleando los filtros disponibles en *Shodan*.

Por otro lado, tal como se mencionaba anteriormente, la *API* permite realizar varios tipos de búsquedas, pero existen algunas limitaciones que restringen el número total de resultados retornados y el uso de protocolos como *HTTPS* y *Telnet*. Para saber las limitaciones que tiene una cuenta concreta a la hora de utilizar la *API* y los *add-ons* que tiene habilitados, se utiliza la función info.

Script: ShodanDeveloperKeyInfo.py

```
#!/usr/bin/env python
import Shodan
def ShodanTest():
    try:
        ShodanKeyString = "API KEY"
        ShodanApi = Shodan.Shodan(ShodanKeyString)
        info = ShodanApi.info()
        for inf in info:
            [illegible] '%s: %s ' %(inf, info[inf])
    except [illegible], e:
        [illegible]

if __name__ == "__main__":
    ShodanTest()
```

En este caso, se pintan por pantalla los detalles de configuración relacionados con la cuenta de usuario que ha creado la instancia de la clase *Shodan.Shodan*. En el caso de que la cuenta de usuario tenga los *add-ons* de *HTTPS* y *Telnet* habilitados, aparecerá lo siguiente:

```
>python simpleShodan py
https: True
 [illegible]acked left: 100
[illegible]
[illegible]
unlocked  True
```

Otra característica muy interesante que tiene la *API* de *Shodan*, es la posibilidad de utilizar "facetas", las cuales permiten generar agrupaciones de datos para su posterior análisis. El uso de las facetas permite obtener información global sobre los registros que se encuentran almacenados en la base de datos de *Shodan*, por ejemplo, sería posible extraer los "n" países que tienen el mayor número de servidores *web* con *Apache* o *NGINX*. Para utilizar esta característica, la función *count* permite obtener el numero total de registros de una busqueda sin retornar los registros encontrados.

Script: ShodanSimpleFacets.py

```
#!/usr/bin/env python
import Shodan
[illegible]ETS
    ([illegible]),
    ('domain',15),
    ('port', 15),
    ('asn', 10),
    ('[illegible]y', 15),

FACET TITLES = {
    'org': 'Top 15 Organizations',
    'domain': 'Top 15 Domains',
    'port': 'Top 15 Ports',
    'asn': 'Top 10 Autonomous Systems',
    'country': 'Top 15 Countries',

try:
    api = Shodan.Shodan("API[illegible]")
    result = api.count("ngnix", [illegible]ACETS
    print 'Total Results: %s\n' % result['total']
   for facet in result['facets'].
        print FACET TITLES[facet
        for term in re[illegible]
            print '%[illegible] %s' % (term['value'], term['count'])
[illegible]
[illegible]t Exception, e
    print 'Error: [illegible]' e
```

[illegible]n el *script* anterior se realiza una busqueda con el filtro "*apache*" y se pintan por pantalla los resultados correspondientes a las facetas que se han incluido en la variable *FACETS*

[illegible]emás de realizar busquedas globales sobre todos los registros almacenados en *Shodan* también [illegible]ste la posibilidad de realizar busquedas concretas contra un segmento de red particular, del mismo [illegible]do que se puede hacer con el filtro *net*, sin embargo, para hacer esto desde la *API* de *Shodan*, se [illegible]uentra disponible la función host la cual admite una dirección *IP* como argumento

Script: ShodanSimpleSearchHost.py

```
# /usr/bin/env python
import Shodan
def ShodanTest():
    try:
        ShodanKeyString = "API K[illegible]"
        [illegible] = [illegible]odan(ShodanKeyString)
        res[illegible] = [illegible] host("188.84.47[illegible]")
        [illegible] res[illegible]
            print '[illegible]' [illegible], results[result]
    [illegible]
        print 'Er[illegible]'

[illegible]
[illegible]
```

Como se puede apreciar en el *script* anterior, se ejecuta una búsqueda en *Shodan* contra la dirección *IP* que se ha especificado como argumento en la función *host* y luego se pintan los resultados por pantalla.

Finalmente, las últimas versiones de la librería, a la fecha de escribir este documento, incluyen nuevas características que permiten acceder en tiempo real a los datos que *Shodan* se encuentra analizando en el momento. Se trata de una característica que permite obtener información adicional que no se encuentra indexada en *Shodan*, ya que toda esta información es filtrada y solamente un conjunto limitado de los datos analizados son indexados en el motor de búsqueda.

No obstante, el uso de esta característica tiene restricciones bastante considerables, ya que en primer lugar, solamente la pueden utilizar aquellos que paguen un plan de suscripción y por defecto solamente retorna un uno por ciento (1%) de la totalidad de los registros que recolecta *Shodan* en *Internet*. En el caso de querer tener acceso a más datos, es necesario solicitar información sobre los precios.

En cualquier caso, tal como mencionaba anteriormente, las características que ofrece *Shodan* son muy valiosas para un *pentester* y tiene un costo bajo.

Script: ShodanRealTime.py

```
#!/usr/bin/env python
import Shodan
import sys
try:
    api = Shodan.Shodan("API KEY")
    for banner in api.stream.ports([443,]):
        if 'opts' in banner and 'pem' in banner['opts'][illegible]
            print banner['opts']['pem'[illegible]


except Exception, e:
    print 'Erro[illegible]
    s[illegible]
```

Con el *script* anterior, es posible recuperar los certificados que están siendo procesados por *Shodan* en el momento y pintar por pantalla la información de cada uno. Además de las funciones explicadas en esta sección, también existen otras funciones y clases que se encuentran incluidas en *Shodan* y que pueden resultar interesantes para el lector. Se recomienda leer la especificación de las funciones y clases disponibles en la *API* desde el siguiente enlace: *https://Shodan.readthedocs.org/en/latest/api.html*

1.5 Ingeniería Social

Un atacante en el mundo real, normalmente no solamente se interesa en atacar sistemas y explotar cualquier tipo de vulnerabilidad, los vectores de ataque utilizados con mayor frecuencia contra empresas y diferentes tipos de organizaciones han sido aquellos que se centran en engañar y manipular a sus empleados para conseguir información confidencial o acceder a zonas restringidas. La ingeniería social se basa en el hecho de que en algunas ocasiones es mucho más fácil y efectivo atacar la "infraestructura humana" que una arquitectura de sistemas debidamente asegurada.

Un ingeniero social, utiliza las herramientas y los conocimientos que tiene a su disposición para conseguir que una persona realice tareas que son acordes a sus intereses particulares, tales como acceder a información, acceder a zonas o espacios restringidos, obtener beneficios de cualquier tipo, etc. Dichos conocimientos se basan en algunas características propias de la psique humana y que por ende, es difícil para una empresa u organización contar con medidas de protección eficaces que permitan minimizar el riesgo y el impacto de un ataque exitoso. Es por este motivo, que muchas de las organizaciones más reconocidas en el mundo de la seguridad informática, tales como *Symantec* han reportado la ingeniería social como una de las principales causas de divulgación de información sensitiva en los últimos años, produciendo perdidas de millones de dólares a empresas publicas y privadas cada año.

Las principales técnicas empleadas por ingenieros sociales, actualmente se encuentran plasmadas y detalladas con un nivel de profundidad bastante aceptable en múltiples libros de texto, sin embargo, uno de los mejores recursos que se puede encontrar en *Internet* sobre este tópico, es el *Social Engineering Framework*.

1.5.1 Social Engineering Framework

El *Social Engineering Framework* es un conjunto de herramientas y recursos útiles para el estudio de las técnicas e innovaciones más recientes en el área de la ingeniería social. La información que incluye el *framework* trata de definir los principios básicos de la ingeniería social así como algunas de las "debilidades" de la psique humana que suelen atacar los ingenieros sociales para conseguir sus objetivos por medio de la manipulación y el engaño. Los pilares de este *framework* se listan a continuación y se incluye un breve resumen para su mejor comprensión. Todos los recursos y

herramientas incluidos en este *framework* pueden encontrarse en la siguiente dirección: *http://www.social-engineer.org/framework/Social_Engineering_Framework*

1.5.1.1 Recolección de Información

Se trata del proceso de recolectar y almacenar de forma ordenada toda la información que sea posible conseguir sobre los recursos humanos del objetivo en cuestión. Para llevar a cabo este proceso, se suelen utilizar herramientas informáticas especializadas para dicho fin, así como también cualquier tipo de dato que pueda ser obtenido en *Internet* o en otros medios tales como diarios, guías telefónicas, mapas sobre la ubicación física del objetivo, entre otras fuentes.

Este es un proceso que requiere habilidad y creatividad para encontrar cualquier tipo de información sobre el objetivo, incluso en los sitios menos usuales, como por ejemplo en los contenedores de basura de una empresa, allí es posible encontrarse con documentos o medios digitales con información importante que no se han destruido por completo y que permiten a un ingeniero social tener los elementos necesarios para elaborar cuidadosamente su ataque.

1.5.1.2 Elicitación

La traducción al castellano más cercana al término "*Elicitation*" es "*Elicitación*". Se trata de uno de los aspectos más importantes en el estudio de la ingeniería social y probablemente, el elemento más importante del *framework*. Este término, describe la habilidad que tienen los ingenieros sociales para generar estímulos que provocan o alteran el comportamiento habitual de otras personas. Esta habilidad, frecuentemente se basa en utilizar la simpatía o producir sentimientos agradables sobre el objetivo, ya que es mucho más fácil extraer información valiosa cuando una persona se siente abierta, confiada y relajada. La *NSA* (*National Security Agency*) de los Estados Unidos de America, define el término "*Elicitation*" como: "La habilidad de extraer información de forma sutil durante una conversación aparentemente normal e inocente". Se trata de una habilidad muy poderosa que funciona bastante bien dado que encaja con algunos principios psicológicos que son destacados en las interacciones humanas, tales como:

- Los humanos tienden a ser amables y cordiales, especialmente con personas desconocidas.
- Los profesionales de cualquier área, quieren parecer bien formados y que dominan sobre los temas tratados en una conversación.
- Muchas personas cuando reciben halagos, suelen ser más abiertas y tienden a divulgar información fácilmente.
- Muchas personas suelen responder rápidamente a las preguntas para parecer preocupados y atentos durante el proceso comunicativo.
- Muchas personas suelen estar más atentas a prestar su ayudar y pueden divulgar información cuando perciben una posible compensación económica, reconocimiento o simplemente agradecimiento.
- Muchas personas buscan la aceptación de otros siendo amables, educados y simpáticos.

de esta técnica, es extraer la mayor cantidad de información de una persona utilizando ... sencillas y mantener el interés en la conversación, para conseguir dicho objetivo, es ... explotar algunos de los aspectos psicológicos anteriormente mencionados y llevar la ... por el camino que le resulte conveniente al atacante.

5...3 Pretextos

... la habilidad de crear un escenario inventado con el fin de persuadir a una persona a ... información o ejecutar alguna acción. Para que dicho escenario pueda dar los mejores ... es necesario que el ingeniero social pueda despertar un sentimiento de confianza en la ... si esto no es posible, como en todas las relaciones humanas, la otra persona no podrá ... persuadida a entregar información o realizar alguna acción.

5...4 Principios Psicológicos y PNL (Programación NeuroLingüística)

... principios psicológicos que son importantes desde el punto de vista de la ingeniería ... que permiten conocer costumbres, modos de pensar, aptitudes y actitudes en la otra ... de estos principios consiste en conocer, mediante el proceso comunicativo, cuáles son ... que la otra persona tiene más desarrollados.

... hay personas que perciben el mundo con mayor agudeza utilizando la vista, mientras ... que lo hacen utilizando el oído o el tacto. Para un ingeniero social es importante saber ... ya que de esta forma puede comunicarse de una forma mucho más eficiente y eficaz ... En realidad se trata de conceptos que son un poco más complejos de lo que ... requieren mucha práctica para poder llegar a dominarlos en conversaciones habituales ... personas.

...do, la *PNL* (*Programación NeuroLingüística*) es un modelo que permite conocer las ... entre los patrones de comportamiento y conducta, teniendo en cuenta variables externas ... contexto y las condiciones en las que se establece la comunicación con otras personas.

... *Jung* escribió: "Todo depende de cómo veamos las cosas y no sobre como son realmente" ... esta frase cobra mucho sentido, dado que se trata de analizar el comportamiento de las ... modos de pensar, la forma en la que expresan sus pensamientos, su lenguaje corporal y ... cultural. En este punto adquiere mucha importancia el concepto de las microexpresiones. ... expresiones faciales que se dan de forma espontánea e inconsciente y que reflejan las ... que experimenta una persona en un momento determinado. Han sido objeto de muchos ... por parte de múltiples investigadores y psicólogos, sin embargo, los trabajos del *Dr. Paul* ... han sido los que han aportado mayores avances en área de las emociones humanas durante ... comunicativo y sobre todo, en la detección del engaño.

... han revelado que las microexpresiones son independientes de factores culturales o ... además ha podido categorizarlas en grupos de emociones positivas y negativas. Para un ... social, el entrenamiento continuo sobre el manejo de las microexpresiones, así como del

lenguaje verbal y corporal son una de sus prioridades, ya que saber controlar estos factores durante una conversación, le permitirán tener cierto control sobre los eventos que se produzcan en la misma.

1.5.1.5 Influencia

Finalmente, este pilar del *framework* consiste en la definición de técnicas de manipulación e influencia sobre otros. Dichas técnicas se basan de forma directa en los conceptos explicados anteriormente, los cuales le brindan las herramientas necesarias a un ingeniero social para elaborar situaciones que le permitan tener cierto nivel de control sobre otras personas.

En este punto, las tácticas definidas en el *framework* son:

- **Reciprocidad**: Le permite a un ingeniero social influenciar a otra u otras personas basándose en el principio psicológico de "reciprocidad", el cual está fuertemente instaurado en la cultura occidental. De esta forma la otra persona tendrá un sentimiento de obligación moral, social, contractual o incluso legal (aunque sea ficticio), obligándole a llevar a cabo alguna acción o simplemente divulgar información. Este principio solemos aplicarlo día a día de manera casi que inconsciente, por ejemplo, cuando dejamos libre un puesto para sentarse a una persona mayor en el autobús o cuando ofrecemos nuestra colaboración en cualquier asunto laboral o académico a un compañero. Aunque se trate de ejemplos bastante comunes, un ingeniero social con buenas habilidades, puede elaborar contextos mucho más sofisticados para explotar dicha condición en la naturaleza de los seres humanos para influenciar el comportamiento de otros.
- **Escasez**: Se trata de una táctica frecuentemente usada para crear un sentimiento de urgencia en un contexto de toma de decisiones, obligando de esta forma a una persona a tomar decisiones que no siempre siguen los intereses particulares de la víctima, sino que buscan satisfacer los intereses del ingeniero social. Se trata de una técnica utilizada con mucha frecuencia por vendedores para hacer que los compradores actúen de forma rápida creando un sentimiento de escasez de un producto o servicio determinado. Este mismo concepto aplica en muchos contextos, tales como la economía, la política, entre otros. Es una táctica muy poderosa que puede ser utilizada para influenciar el comportamiento de las personas y manipular sus decisiones en función a los intereses del ingeniero social.
- **Autoridad**: Un ingeniero social puede obligar a otra persona a realizar algún tipo de acción o divulgar información basándose en la legitimidad que le otorga un puesto de mando en una empresa, estatus social o autoridad legal. Dicha autoridad puede ser real o ficticia y suele ser parte del contexto elaborado por el ingeniero social para cumplir sus objetivos. En cualquier caso, la elección del tipo de autoridad que puede asumir como propia un ingeniero social dependerá directamente de los incentivos y características del objetivo. Por ejemplo, si el objetivo es un militar o un policía, probablemente sea mucho más eficiente elaborar un contexto de autoridad jerárquica simulando un alto mando para influir sobre el comportamiento y las decisiones de dicha persona. Otro ejemplo del uso de esta táctica, que suele ser muy eficiente, es por medio de la autoridad legal, donde un ingeniero social puede alegar legitimidad en el acceso a información sensible, basándose en argumentos legales que aunque puedan parecer legítimos, frecuentemente son inválidos.

Gustos. Muchas personas suelen ser más propensas a ser influenciadas por objetos o personas que son acordes a sus gustos. Si el ingeniero social conoce estos detalles sobre su objetivo, puede elaborar un contexto controlado que le permita acceder a información sensitiva o hacer que el objetivo lleve a cabo alguna acción que cumpla con los intereses del ingeniero social. Los gustos de una persona, evidentemente no son una medida lineal y pueden cambiar considerablemente entre diferentes personas o incluso entre modas y épocas, por este motivo, es necesario conocer los gustos y aficiones del objetivo para crear el contexto adecuado y poder influenciar su conducta.

Estos han sido, a grandes rasgos, los pilares fundamentes sobre los que se basa el framework, sin embargo, también existen herramientas tanto físicas como automatizadas que facilitan su aplicación.

1.5.1.6 SET (Social Engine Toolkit)

SET es una herramienta escrita en *Python* que se enfoca principalmente en la explotación del "factor humano" y ha sido específicamente diseñado para soportar los conceptos teóricos y prácticos expuestos en el *Social Engineering Framework*.

El objetivo de esta herramienta es el de servir como punto de apoyo de un ingeniero social soportando varios tipos de ataques del tipo "*Client-Side*" con el único fin de comprometer el espacio de trabajo del mayor número de usuarios posible.

Algunos de los tipos de ataque que soporta se listan a continuación

- *Spear-Phishing Attack Vector.*
- *Infectious Media Generator.*
- *Java Applet Attack Vector.*
- *Metasploit Browser Exploitation.*
- *Credential Harvester Attack.*
- *Tabnabbing Attack.*
- *Man Left in the Middle Attack.*
- *Web Jacking Attack.*
- *Multi-Attack Web Vector.*
- *QRCode Attack.*

Además de estos tipos de ataque, también le permite a los desarrolladores crear módulos que se puedan incluir directamente en la herramienta.

1.5.1.6.1 Desarrollo de Módulos en SET

Aunque se trata de una herramienta que permite establecer rápidamente distintos vectores de ataque, la ventaja de crear módulos propios o incluso reutilizar otros escritos por terceros, es una de las características más potentes que tiene esta herramienta.

El procedimiento es sencillo, solamente es necesario incluir en el directorio "<SET INSTALL>/modules" uno o varios ficheros *Python* (*.py) que contendrán las rutinas que serán ejecutadas por el modulo. Todos los *scripts* incluidos en dicho directorio, serán cargados en *SET* bajo la opción "*Third Party Modules*".

A continuación se incluye un modulo simple que se puede cargar en *SET*. Nótese el uso de las clases y utilidades ya incluidas en la herramienta.

Script: setSimpleExample.py

```
[illegible]
[illegible]
[illegible]

MAIN="Simple Module"
AUTHOR="Adastra"
def [illegible]
        [illegible]ck("http://thehackerway.com","80","thw/")
```

En el *script* anterior, la función *main* es obligatoria ya que será la función que *SET* ejecutará automáticamente. Por otro lado, el modulo *core* contiene todas las utilidades y funciones que incluye el framework y que un desarrollador de *Python* puede utilizar desde sus propios *scripts*.

En este caso concreto, solamente se utiliza la función *core java applet attack* la cual se encarga de clonar un sitio, iniciar un servidor *web* con *Metasploit* en el puerto 443 y finalmente iniciar un *handler* de *metasploit* para recibir cada *shell* reversa de las victimas.

Además de la función *core java applet attack*, existen algunas otras que permiten combinar varios ataques desde un único modulo. Un ejemplo un poco mas concreto y que extiende las funcionalidades de *SET* se puede apreciar en el siguiente *script*

Script: setFunctionsExample.py

```
#![illegible]
[illegible]import *
import sys
MAIN="Multi-use Mo[illegible]"
AUTHOR="Adastra"
def main()
        site cloner("[illegible]p://thehackerway.com","thw/", "")
        print check os()
        print meta path()
        print meta database()
        ipAddress = grab ipaddress()
        [illegible]
        [illegible]
                [illegible]
      [illegible]cp", ipAddress, "31337")
      [illegible]
      [illegible]_web_server("thw/")
      [illegible]_dns()
```

El *script* anterior utiliza varias de las funciones disponibles en *SET* y que estan a disposición de cualquier desarrollador. En primera instancia, se ejecuta la función *site_cloner* para extraer los contenidos *web* de un dominio determinado para posteriormente poder crear un "*fake site*". Las siguientes funciones se encargan de realizar tareas variadas, tales como determinar la version del sistema operativo, obtener información relacionada con el directorio de instalación de *Metasploit Framework* y determinar si una la direccion *IP* local es valida. La funcion *generate_shellcode* se encarga de generar un *shellcode* ejecutando la utilidad *msfvenom* de *Metasploit Framework*. La función *generate_random_string* toma el rango de valores entre 1 y 35 para generar una cadena aleatoria y finalmente, las funciones *start_web_server* y *start_dns* se encargan de iniciar un servidor *web* y un servidor *DNS* en la maquina local respectivamente.

Algunas de las funciones disponibles en *SET* para utilizarlas desde cualquier modulo, se pueden apreciar a continuación.

- ***site_cloner(website, exportpath, *args)***: Permite clonar un sitio *web* y almacenarlo en el directorio especificado. Si se ejecuta lo siguiente: *core.site_cloner ("http://www.google.com", "google/")* el contenido del sitio *http://www.google.com* sera almacenado en el directorio "google".
- ***meta_path()***: Permite conocer el directorio de instalacion de *Metasploit Framework* (el que se encuentra definido en el fichero de configuracion "*set_config*").
- ***check_pexpect()***: Verifica si la libreria *Pexpect* se encuentra instalada en el sistema.
- ***check_beautifulsoup()***: Verifica si la libreria *BeautifulSoup* se encuentra instalada en el sistema.
- ***update_set()***: Actualiza la version de *SET*.
- ***update_metasploit()***: Actualiza la version de *Metasploit Framework*.
- ***help_menu()***: Enseña el menu de ayuda contextual de *SET*.
- ***date_time()***: Retorna la fecha y la hora.
- ***generate_random_string(low, high)***: Genera una cadena aleatoria entre *low* y *high*.
- ***start_web_server(dir)***: Inicia un servidor *web* en el directorio especificado.
- ***start_web_server_unthreaded(dir)***: Inicia un servidor *web* en el directorio especificado en un único hilo de ejecución.
- ***upx(file)***: Codifica el fichero especificado utilizando *UPX*.
- ***show_graphic()***: Enseña una opción aleatoria del menu principal.

Si el lector se encuentra interesado en profundizar aun mas en el desarrollo de modulos en *SET* se recomienda leer la sección correspondiente en el manual de usuario alojado en el directorio "*<SET_INSTALL>/readme/User_Manual.pdf*". Además, tambien es muy instructivo leer el código fuente del proyecto, más concretamente los ficheros "*<SET_INSTALL>/src/core/setcore.py*" y "*<SET_INSTALL>/src/core/set.py*" los cuales contienen las funciones que se han descrito anteriormente, asi como otras que estan disponibles y que pueden ser de utilidad para el desarrollador.

1.5.2 Maltego

Maltego es una herramienta diseñada con el fin de recolectar y estructurar diferentes *ítems* de información de una forma visual muy fácil de comprender. Los *ítems* pueden ser dominios en *Internet*, sitios *web*, direcciones de correo electrónico o incluso información personal. Es una herramienta muy valorada y utilizada en las primeras etapas de recolección de información ya que se permite ejecutar búsquedas por datos personales o información relacionada con una organización concreta. Para realizar estas búsquedas, es necesario ejecutar una transformación, que es simplemente una acción en *Maltego* que permite especificar el tipo de búsqueda que se debe realizar. Por ejemplo, se puede especificar que se realicen búsquedas en servidores *SMTP* públicos por una dirección de correo electrónico concreta, búsquedas por números de teléfono asociados a una persona o simplemente todas las transformaciones disponibles en *Maltego* para recuperar la mayor cantidad de información posible.

Si se quisiera indagar sobre la información pública de una persona y cualquier otro detalle que pueda ser de interés, partiendo de su cuenta en twitter, se puede crear un nuevo proyecto en *Maltego* y seleccionar el *nodo* "Personal" en la paleta ubicada a la izquierda, tal y como se enseña en la siguiente imagen.

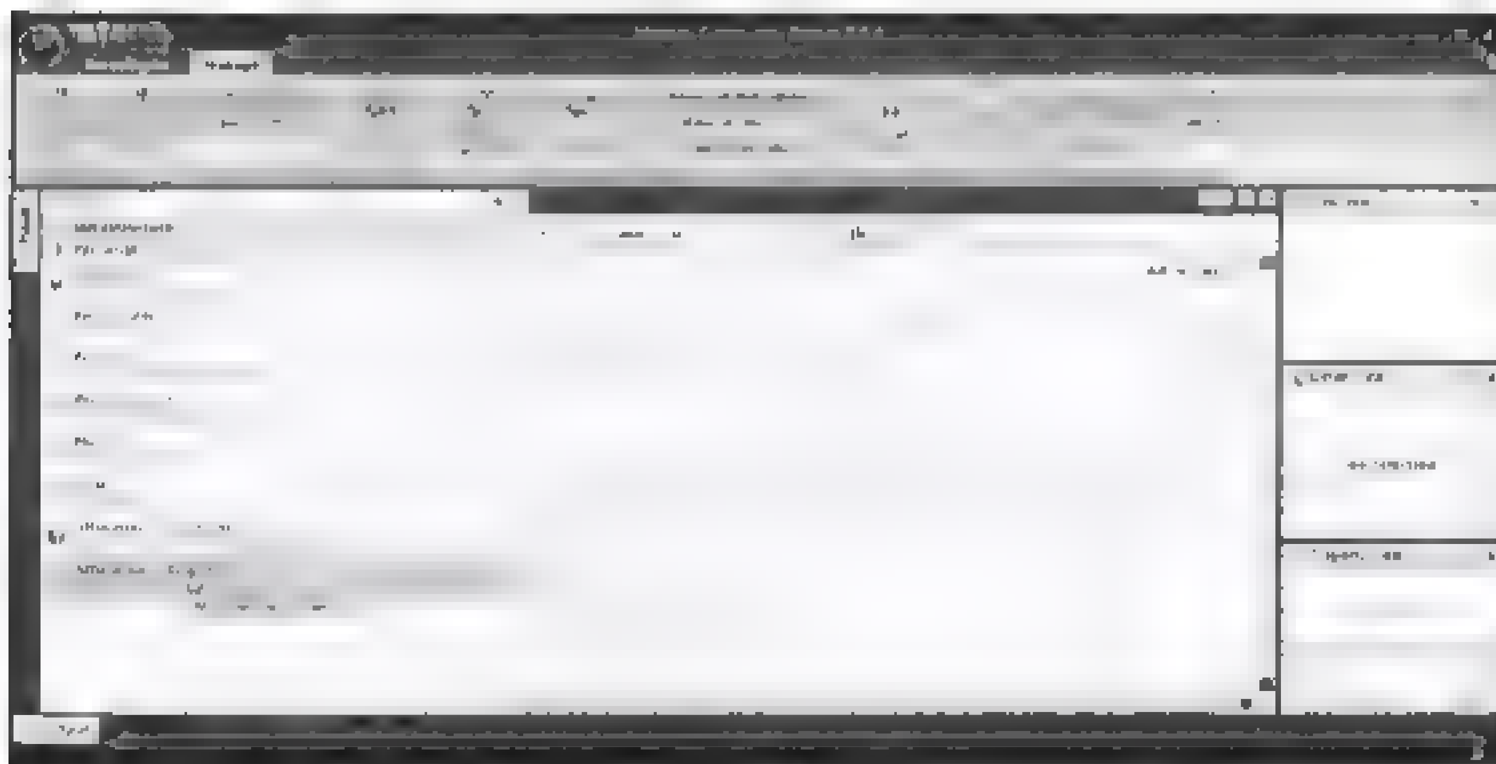


Imagen 01.01: Creación de un objeto *"Affiliation Twitter"* en *Maltego*.

Posteriormente, solo es necesario ingresar la información de la cuenta a buscar y sobre el elemento creado en el panel central, ejecutar cualquiera de las transformaciones disponibles pulsado click derecho sobre el objeto.

Las imágenes 01.02 y 01.03 enseñan las relaciones creadas después de ejecutar las transformaciones sobre la cuenta en *twitter* deseada.

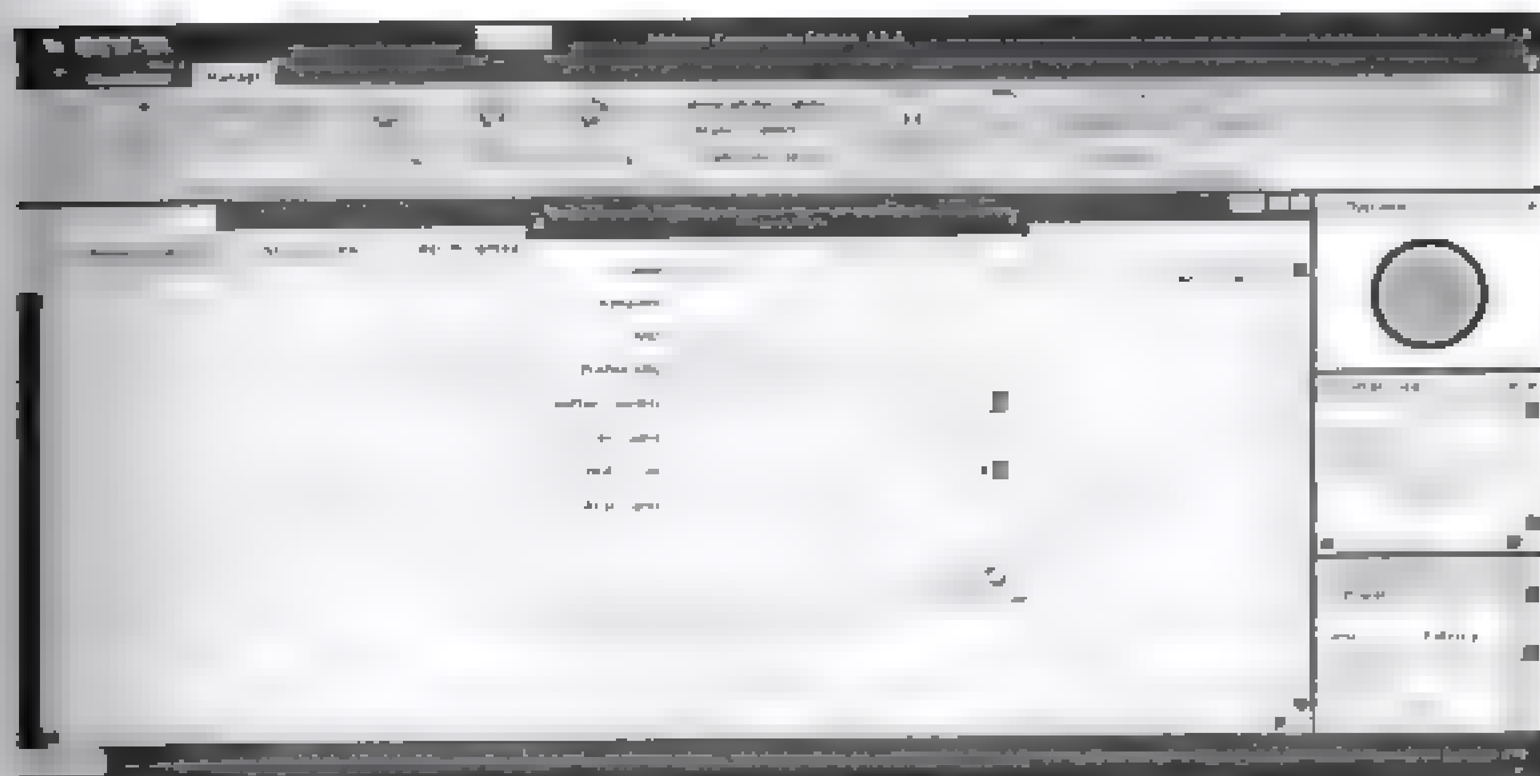


Imagen 01.02. Establecimiento de los datos de una cuenta en *Twitter*



Imagen 01.03. Resultado de las transformaciones ejecutadas desde *Maltego*.

... se trata de un ejemplo muy sencillo, demuestra la potencia y simplicidad de esta ...enta, que no solamente se limita a la busqueda y recuperación de información de personas y ...aciones, sino que tambien permite realizar búsquedas sobre la infraestructura de un sistema, ... web, dominio o una *URL* recolectando toda la informacion disponible por medio de las ...rmaciones definidas.

La imagen 01.04 enseña los resultados de ejecutar todas las transformaciones disponibles en *Maltego* para un sitio *web* determinado. Como se puede apreciar, se han encontrado datos relacionados con la infraestructura del sitio *web* que pueden ser muy útiles para elaborar un vector de ataque contra el sitio *web* en cuestión.

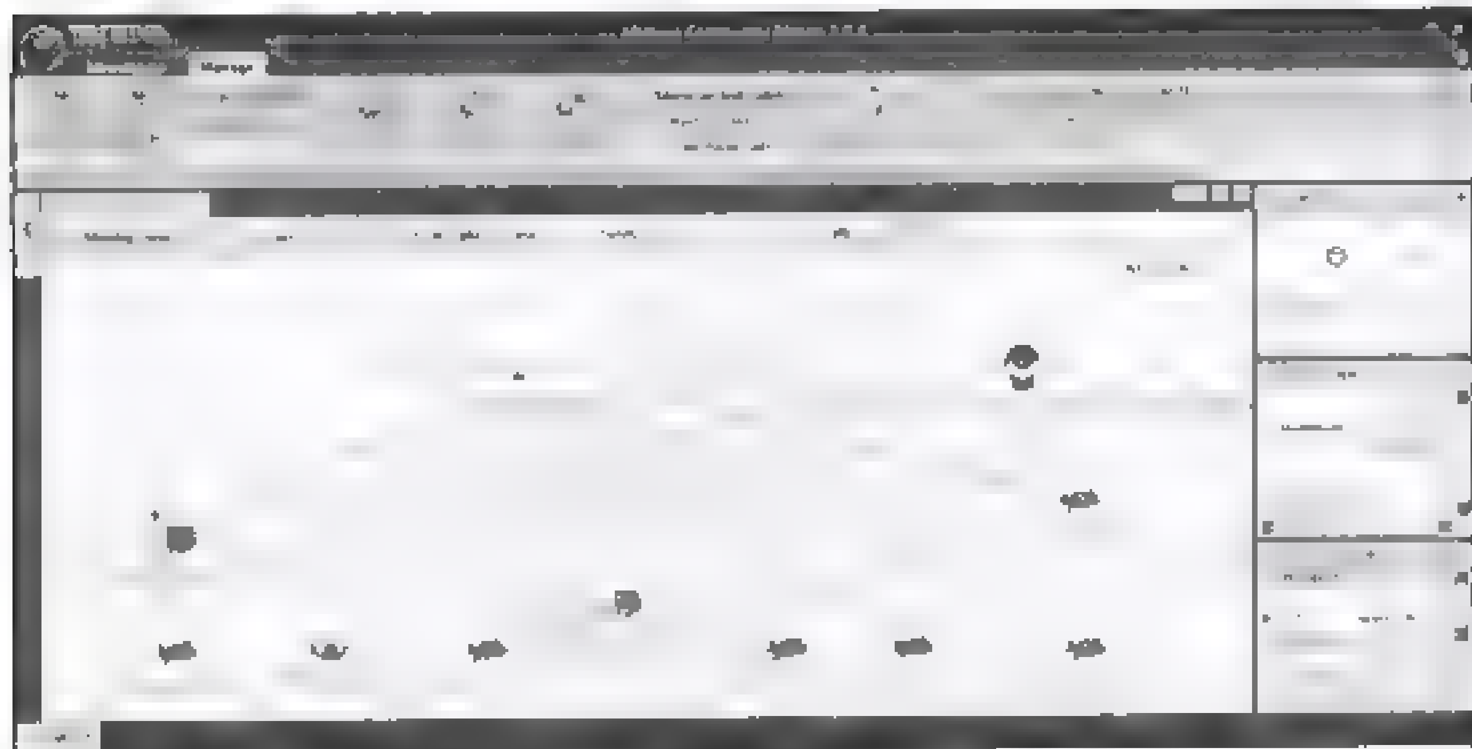


Imagen 01.04: Información sobre la infraestructura de un sitio *web* utilizando *Maltego*

1.5.2.1 Canari Framework

Aunque las transformaciones incluidas en *Maltego* son suficientes en la mayoría de los casos, es posible crear transformaciones propias o utilizar otras escritas por terceros. En el caso de *Canari Framework*, un atacante cuenta con una potente herramienta para crear transformaciones que pueden integrarse fácilmente en *Maltego* utilizando *Python*. Una de las características más llamativas y potentes de *Canari* es que utilizando su *API* es posible abstraer muchas de las complejidades que implica desarrollar transformaciones en *Maltego* y le permite a un desarrollador enfocarse directamente en la lógica que desea implementar para extraer, filtrar y enseñar información útil.

1.5.2.1.1 Instalación de Canari Framework

Para instalar *Canari*, es necesario tener una versión de *Python* superior a la 2.7 y tener instalada la librería *setuptools* para ejecutar *easy_install* cómodamente.

```
[illegible]
```

Con el comando anterior, se instalan automáticamente todas las dependencias necesarias para poder ejecutar *Canari Framework* y comenzar a crear transformaciones en *Maltego*.

El proceso de instalación es muy simple y a la fecha de escribir este documento, *Canari* es soportado en sistemas *Windows*, *Linux* y *MacOS*. Si el lector tiene cualquier dificultad a la hora de instalar

[illegible] se [illegible] la [illegible] la sección correspondiente de documentación oficial en el siguiente enlace: [illegible]www.canariproject.com/installing-canari

1.5.2.1.2 Transformaciones locales y remotas en Canari Framework

[illegible] de los conceptos clave a la hora de trabajar con *Canari Framework* es conocer el proceso de [illegible] y ejecución de transformaciones locales y remotas. Las transformaciones son funciones [illegible] se ejecutan directamente contra una instancia de *Maltego* y los resultados de la ejecución de [illegible] funciones son notificados por medio del canal *stdout* (salida estándar del sistema).

[illegible] transformaciones locales y remotas funcionan exactamente igual, la diferencia es simplemente que [illegible] transformaciones remotas pueden ser alojadas en un servidor de distribución de transformaciones [illegible] *Distribution Server, TDS*) que es gestionado y administrado por *Paterva* (organización [illegible] desarrolla y mantiene *Maltego*). Las ventajas de las transformaciones remotas son evidentes, en [illegible] lugar son fáciles de distribuir sobre múltiples instancias de *Maltego*, lo que le permite a los [illegible] desarrolladores compartir sus transformaciones con otros usuarios. Además, *Paterva* se encarga de [illegible] los derechos de autor sobre las transformaciones que son subidas al *TDS*, protegiendo de [illegible] forma las transformaciones creadas por cada desarrollador. Esto es muy útil en el caso de querer [illegible] transformaciones y distribuirlas fácilmente a cualquier cliente.

1.5.2.1.3 Desarrollo de Transformaciones con Canari Framework

[illegible] de que *Canari* es instalado, la utilidad por línea de comandos *canari* permite, entre otras [illegible] crear transformaciones muy fácilmente con el argumento create-package.

```
[illegible]
[illegible] to the Canari transform package wizard.
[illegible] you like to specify authorship information? [Y/n]: Y
[illegible] description [illegible], Canari [illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible] information? [Y/n]: Y
[illegible] skeleton in canariTransform
[illegible] directory canariTransform
[illegible] directory [illegible]
[illegible]
[illegible]
[illegible] canariTransform/src/canari[illegible]
[illegible] canariTransform/src/canari[illegible]
[illegible] iTransform/src/canari[illegible]
[illegible]
[illegible]
[illegible] canariTransform/resources/external
[illegible] directory [illegible]
[illegible] file canariTransform/ [illegible]
[illegible] file canariTransform/setup.py
[illegible]
[illegible]
[illegible]
```

```
creating file canariTransform/src/ca[illegible]py
creating file canariTransform/src/canariTransform[illegible]py
creating [illegible]py
creating file canariTransform/src/canariTransform/[illegible]py
creating [illegible]py
creating [illegible]/canariTransform
[illegible]
creating file canariTransform/src/canariTrans[illegible]transforms/__init__.py
creating file canariTransform/src/canariT[illegible]transforms/helloworld.py
creating file canariTransform/src/canariTransform/transforms/common/__init__.py
creating file canariTransform/src/cana[illegible]py
done
```

Al ejecutar el comando anterior, se ha solicitado información sobre la transformación a crear y el autor. En el caso de crear un esquema base, se instalan una serie de paquetes y ficheros *Python* que permitirán crear las transformaciones rápidamente.

Al mirar detenidamente en el *log* generado por la herramienta, se puede ver que se ha creado un fichero con nombre *helloworld.py* y en dicho fichero se pueden comenzar a crear las transformaciones que posteriormente se ejecutarán desde *Maltego*.

El fichero *canariTransform/src/canariTransform/transforms/helloworld.py* contiene la información suministrada anteriormente y dos funciones con el decorador *@configure*, la primera de ellas es *dotransform* la cual recibe como argumentos la petición y la respuesta. Se trata de la función principal de la transformación ya que es invocada cuando desde la *interfaz* visual de *Maltego* se intenta aplicar la transformación en cuestión. La segunda función es *onterminate* y es invocada cuando la transformación ha finalizado su ejecución, con lo cual es útil principalmente para realizar tareas de limpieza y liberación de recursos.

1.5.2.1.4 Instalación de Transformaciones en Maltego

Para instalar la transformación y poder utilizarla correctamente desde la interfaz de usuario de *Maltego* es necesario ejecutar la utilidad *canari* con el argumento *install-package* sobre el directorio donde se encuentra el código fuente (en este caso, en el directorio *src*).

```
>canari install-package canariTransform
Writing canari.config to /home/adastra/.canari
[illegible] to /home/adastra/.cana-
ri/canariTransform.conf
Updating /home/adastra/.canari/canari.conf
Looking for transforms in canariTransform.transforms
[illegible]
from canariTransform.transforms.helloworld ...
```

Además de lo anterior, también es necesario ejecutar el *script setup.py* que se ha creado de forma automática en el momento de crear el paquete e ingresar el argumento *install*. Es importante realizar estos dos pasos para que *Maltego* ejecute correctamente la transformación que se ha creado y pueda encontrar las dependencias necesarias.

```
>python setup.py install
```

... estos dos pasos, es suficiente para tener disponible la transformación desde la *interfaz* de ...go y solamente sobre los *items* establecidos en el decorador *@configure* tal como se ha ...cado anteriormente.

...2.2 Transformaciones de Shodan en Maltego

...e con *Canari Framework* es posible crear transformaciones personalizadas, también existe ...idad de utilizar transformaciones desarrolladas por terceros y en el caso de *Shodan*, hay ...serie de transformaciones que pueden ser utilizadas para realizar búsquedas directamente contra ... desde *Maltego*. Las ventajas saltan a la vista, en primer lugar se puede estructurar toda la ...ción disponible sobre un objetivo desde la interfaz de *Maltego* y posteriormente realizar ...uedas en *Shodan* utilizando las entidades cargadas en la paleta. El enlace para descargar las ...formaciones es el siguiente: *https://maltego.Shodan.io*

...alar esta transformación o cualquier otra creada por un tercero, es necesario seleccionar ...*Maltego* el menú *Manage* y seleccionar la opción *Import entities* desde donde se pedirá un ...chero con extensión *mtz* o *mtgx* con todos los recursos necesarios para la creación de las entidades.

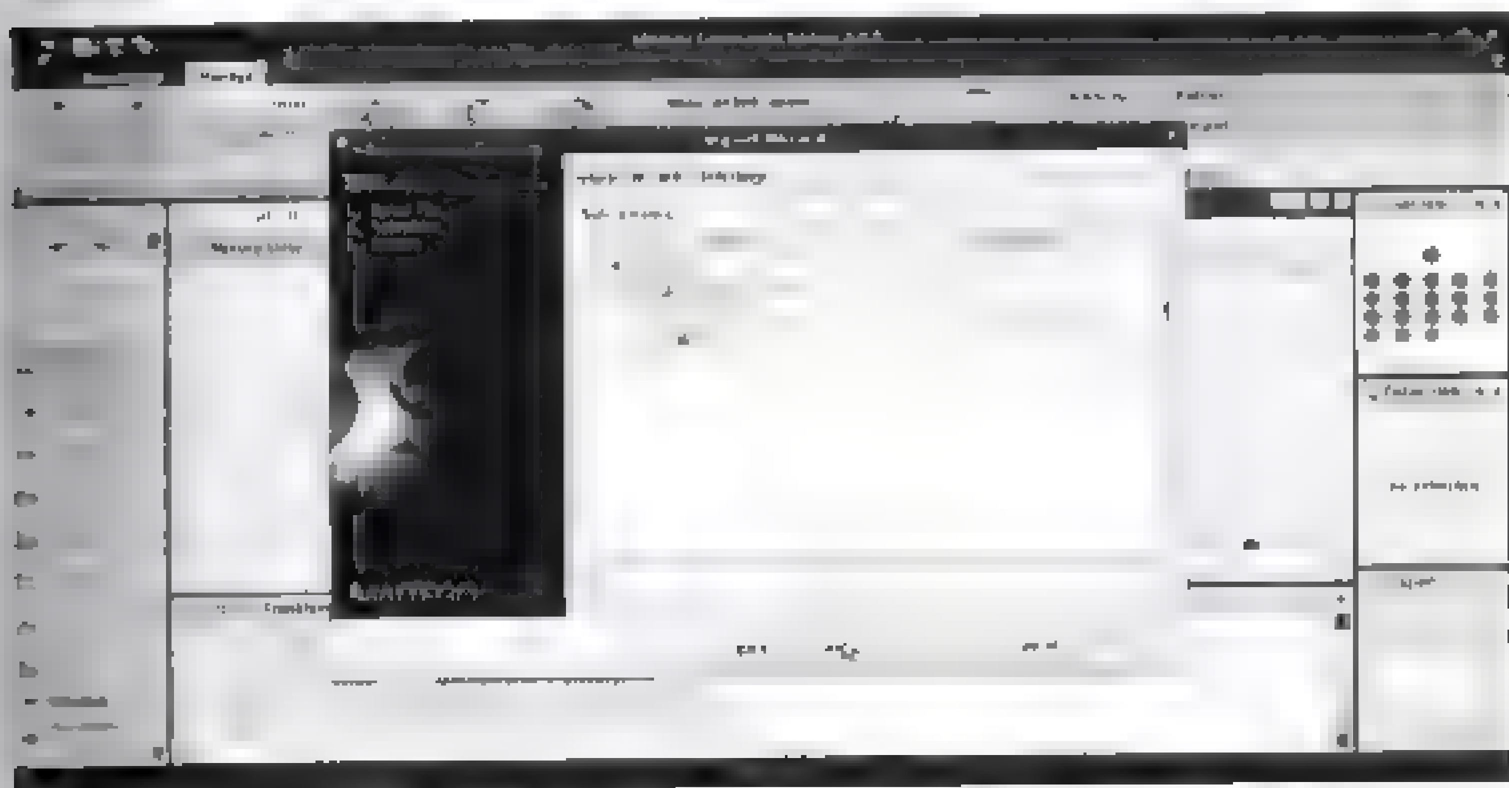


Imagen 01.05: Importación de Entidades en *Maltego*

...ez se han instalado las entidades, el siguiente paso es gestionar las transformaciones y para ...es necesario seleccionar en el menú *Manage* la opción *Discover Transforms*.

...s pasos anteriores, se crean dos elementos nuevos en la paleta de la izquierda, los cuales ...en servicios en *Internet* y *exploits* que se pueden buscar en múltiples repositorios desde ... Además de lo anterior, se puede cargar en el panel un elemento de tipo *Phrase* y comenzar ...búsquedas.

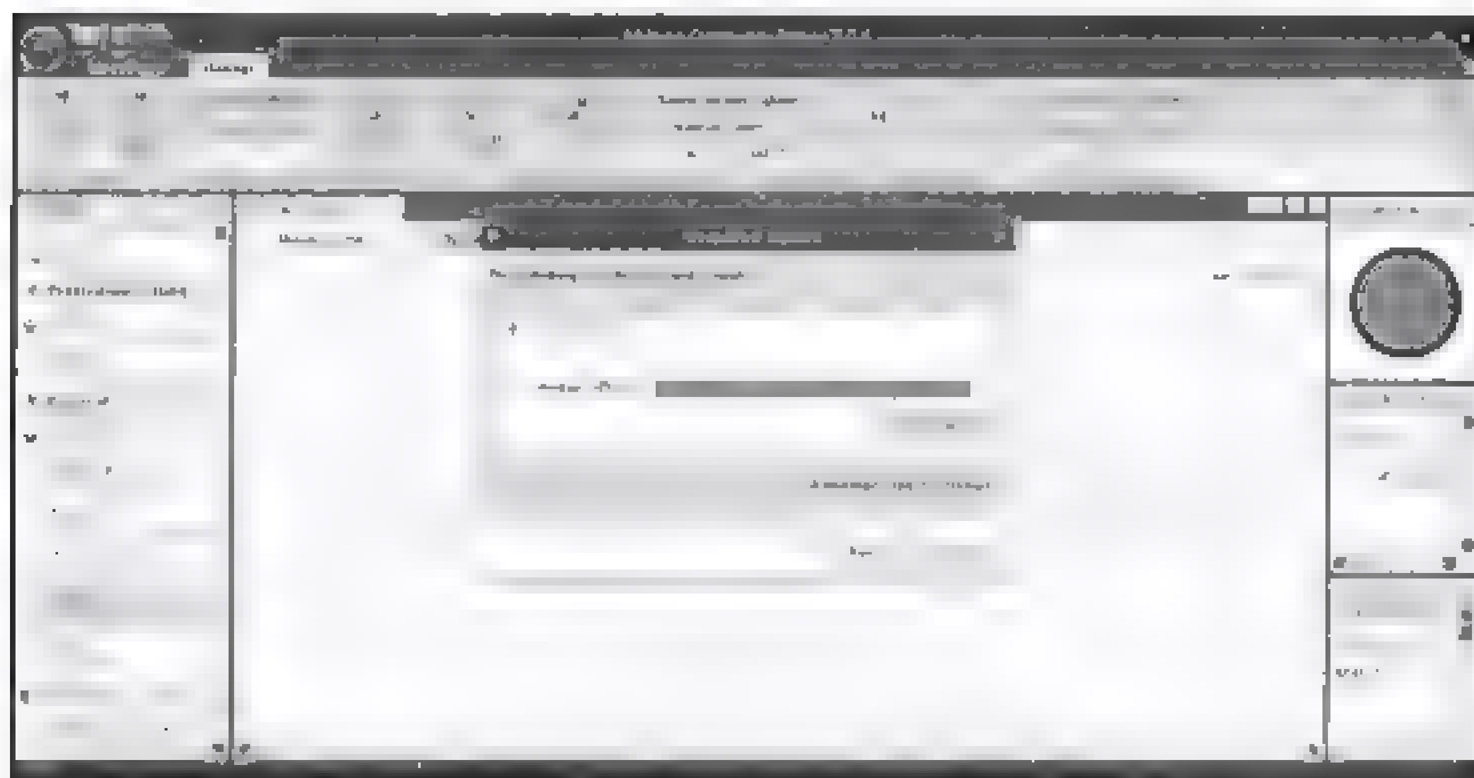


Imagen 01.06: Ejecución de la transformación *scanInShodan* (1ª parte)

Después de ingresar una *Developer Key* válida, es posible ejecutar la transformación y recuperar un conjunto de entidades que representan los mismos resultados que se podrían obtener desde un *script* en *Python* o el sitio *web* de *Shodan*, con la diferencia de que en esta ocasión se generará un conjunto de entidades nuevas en la paleta de *Maltego* a las que es posible seguir aplicando otras de las transformaciones disponibles en la herramienta.

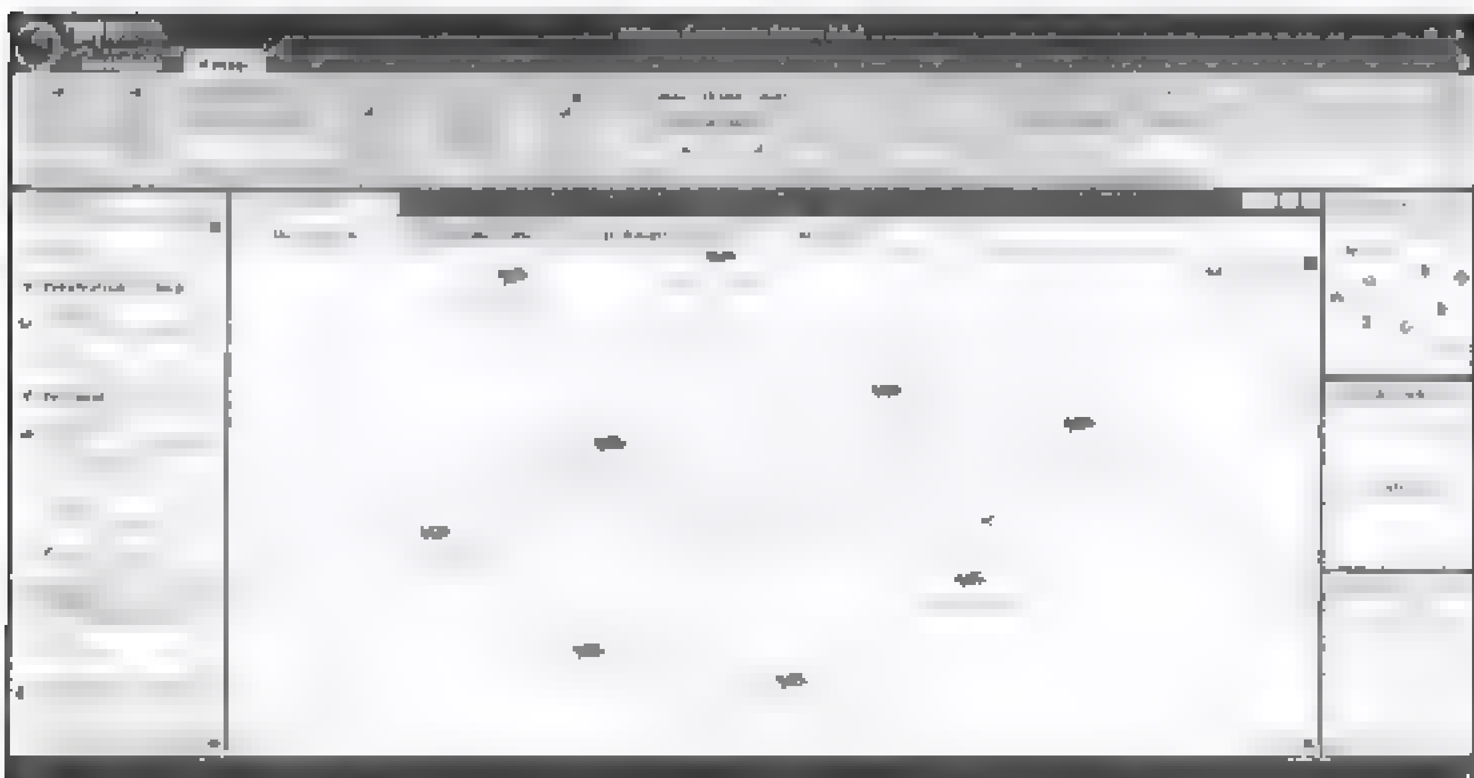


Imagen 01.07: Ejecución de la transformación *scanInShodan* (2ª parte)

La combinación de *Maltego* con *Canari Framework* y las transformaciones que brinda *Shodan*, constituyen un marco de trabajo muy potente que en las manos de un atacante con la firme intención de hacer daño, puede representar un riesgo muy serio para organizaciones y empresas de todo tipo, especialmente cuando el objetivo del ataque es la infraestructura humana de la organización.

1.6 Análisis de Metadatos con Python

Los *metadatos* constituyen una parte importante de cualquier documento y contienen información adicional sobre las fechas de creación/edición, autor, herramienta utilizada para la creación del documento y otra información que puede ser útil de cara a un análisis forense. Han sido muchos los casos en los que se han utilizado los *metadatos* de un documento para capturar atacantes en *Internet* o incluso auténticos criminales. Probablemente el caso más conocido ha sido el caso del asesino en serie conocido como *BTK* (*Bind, Torture and Kill*), al que se le atribuían varios asesinatos entre los años 1974 y la década de los 90 en los alrededores de la ciudad de *Wichita, Kansas*. Fueron más de 30 años en los que la policía no tenía pista alguna sobre el autor de los asesinatos, el cual enviaba cartas a la policía y a las agencias de noticias locales en las que daba instrucciones específicas sobre el procedimiento utilizado para asesinar a sus víctimas.

En el 2004, el caso se encontraba en un punto muerto y aunque se contaba con el *ADN* del asesino, ninguna de las pruebas realizadas con los sospechosos y otros habitantes de la zona eran coincidentes. Ese mismo año, *BTK* envió un mensaje a la policía en el que preguntaba si era posible rastrear su identidad si mandaba un *CD*; la policía en un intento por obtener más pistas, contestó que para ellos era imposible conocer su identidad o su ubicación solamente con un *CD*, aunque en dicho año ya se contaba con las herramientas necesarias para extraer *metadatos* e información adicional sobre cualquier tipo de documento. *BTK* envió un *CD* que contenía un único fichero en formato *RTF*, el cual contenía varios *metadatos* que fueron vitales para localizar y capturar a *BTK*. En los *metadatos* se indicaba el campo "autor" con el valor "*Dennis*" y había una asociación con la iglesia luterana de *Wichita*, lo cual permitió a la policía limitar enormemente la búsqueda. *Dennis* era uno de los diáconos de dicha iglesia, el autor del documento y de los asesinatos.

En el 2005 fue arrestado por la policía de *Wichita* y ese mismo año, confesó los crímenes cometidos delante del juez y el jurado. El caso de *Dennis Rader* no ha sido el único caso en el que agencias como la policía han utilizado los *metadatos* para encontrar el autor de un documento, otro caso más reciente y desde luego, mucho menos escatológico, fue el caso de uno de los integrantes de *Anonymous* llamado *Alex Tapanaris*, el cual fue detenido por la policía griega tras examinar los *metadatos* de uno de los documentos del grupo *hacktivista* en el que se explicaba la operación *Greece*, la cual consistía en una serie de ataques de denegación de servicio distribuido contra varias empresas y multinacionales. La policía logró determinar que *Alex Tapanaris* era el autor de dicho documento tras examinar los *metadatos* incluidos en el fichero *PDF* y unos días después fue

En los últimos años, el uso de los *metadatos* para determinar el autor de un documento o incluso su ubicación geográfica, ha sido una forma bastante empleada por la policía y por los delincuentes para acceder a información sensible que evidentemente, los autores no desearían exponer públicamente.

En *Python* existen varias librerías que pueden ser utilizadas para automatizar la extracción de dicha información a continuación se explica el uso de algunas de ellas.

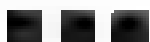
1.6.1 Análisis de metadatos en documentos PDF con PyPDF2

PyPDF es una librería que permite manipular documentos *PDF* así como la posibilidad cifrar o descifrar documentos y desde luego, extraer sus *metadatos* incluidos. Es una librería cuyo desarrollo ha sido abandonado por su autor, sin embargo se ha creado una bifurcación del proyecto para continuar con el desarrollo de nuevas funcionalidades y la solución de problemas existentes en la versión original. Dicha bifurcación ha dado lugar a la librería *PyPDF2*, la cual implementa algunas mejoras con respecto a su predecesora en términos de calidad de código y funcionalidades añadidas. Para descargar la librería es posible hacerlo automáticamente con la utilidad *pip* clonando el repositorio *GIT* o descargando manualmente la última versión estable la cual se encuentra disponible en el siguiente enlace: https://pypi.python.org/pypi/PyPDF2

El uso de la clase *PdfFileReader* permite obtener los *metadatos* de un documento determinado por medio del método *getDocumentInfo*.

```
>>> [illegible]
[illegible] v.1500 32 bit (Intel)] on
[illegible]
[illegible]
>>> from PyPDF2 import PdfFileReader, [illegible]
>>> def [illegible]
        [illegible]
'rb'))
        [illegible]
        [illegible]
            [illegible] + docInfo metaItem
>>> [illegible]
[+] [illegible]
[+] [illegible]
[+] /Creator:[illegible]
[+] /CreationDate[illegible]+02'00'
>>>
```

En este caso, la función *printMeta* crea una instancia de la clase *PdfFileReader* con el documento *PDF* a manipular. Posteriormente, la función *getDocumentInfo* se encarga de retornar un diccionario con todos los *metadatos* del documento. En este caso concreto, se ha utilizado el fichero PDF que ha sido utilizado por la policía griega para capturar a *Alex Tapanaris*. Como se puede apreciar, aparece el nombre completo del autor, la fecha de creación y la herramienta utilizada para su edición. Se trata de un ejemplo simple que puede ser utilizado contra cualquier documento *PDF* y que demuestra el uso de un pequeño conjunto de las funcionalidades incluidas en *PyPDF2*.



1.6.2 Análisis de metadatos EXIF con PIL (Python Imaging Library)

[illegible] (*Exchange Image File Format*) es una especificación que indica las reglas que deben seguirse [illegible] la hora de almacenar ficheros cuyo contenido son imágenes o audio. Esta especificación es [illegible]pecialmente interesante debido a que es aplicada por varios dispositivos tales como *smartphones* o cámaras digitales.

[illegible] los *metadatos* que se pueden incluir en grabaciones de audio y fotografías digitales, se [illegible]entran las coordenadas *PDF* con la ubicación exacta de donde se han creado. Nuevamente, [illegible] tipo de información ha sido muy útil para organismos como la policía a la hora de localizar [illegible]nales o incluso terroristas, pero desafortunadamente, también ha sido utilizada por delincuentes [illegible] perfilar a sus víctimas.

[illegible]mplo de esto, son las imágenes que los usuarios suelen subir a redes sociales como *Facebook*, [illegible] o *Tuenti*, si dichas imágenes tienen información sensible como las coordenadas *PDF*, para [illegible]cuente será mucho más fácil abordar a su víctima físicamente o conocer sus movimientos. [illegible]ema concreto se ha reportado principalmente en los *metadatos EXIF* generados por [illegible]tivos *IPhone 4S*, sin embargo, las versiones más recientes ya no incluyen información [illegible] como las coordenadas *PDF* donde se ha tomado una fotografía digital con la cámara del [illegible]

[illegible] librería en *Python* que permite el procesamiento y manipulación de imágenes. Permite [illegible] *metadatos EXIF* incluidos en un documento muy fácilmente utilizando la función *_getexif* [illegible] de *metadatos* son especialmente extensos y pueden cubrir varias páginas de texto, por este [illegible]casiones es mejor buscar solamente aquellos *metadatos* que resulten interesantes, como [illegible] el campo *GPSInfo*.

[illegible]Metadata.py

```
[illegible]ags import TAGS
[illegible]

[illegible]


[illegible]ge.open(imgFileName)
[illegible]_getexif()



[illegible]ue) in info.items():
    decoded = TAGS.get(tag, tag)
    [illegible]decoded] = value
  [illegible]S = exifData['GPSInfo']
  [illegible]GPS
   print '[*] ' + imgFi[illegible]e + ' GPS Data: '+exifGPS



[illegible]uestra.png')
```

El *script* anterior, se encarga de invocar a la función *testForExif* enviando como argumento la ruta de una imagen. Posteriormente se utiliza la *API* de *PIL* para abrir el documento y extraer los *metadatos EXIF* de la imagen. En el caso de que en los *metadatos* de la imagen se incluya un campo con información sobre las coordenadas *PDF*, se pinta por pantalla el nombre del fichero y las coordenadas extraídas.

Se trata de un ejemplo bastante simple para explicar fácilmente el uso de *PIL* para la extracción de *metadatos EXIF*, pero evidentemente, un atacante podría utilizar otras librerías como *Requests* y *Mechanize* para extraer dinámicamente las imágenes de un sitio *web* y posteriormente aplicar el mismo procedimiento que se ha visto en el *script* anterior para extraer las coordenadas *PDF* de un listado de imágenes.

El uso de *Requests*, *Mechanize* y otras librerías enfocadas a entornos *web*, se verá en el siguiente capítulo de este documento.

Capítulo II
Escaneo, enumeración y detección de vulnerabilidades

...a información recolectada anteriormente, el atacante tiene suficiente para comenzar con ... de escaneo, enumeración e identificación de posibles vulnerabilidades. En esta etapa, un ...e intentará determinar cuáles servicios se encuentran en ejecución en la máquina objetivo. La ...ción recogida en esta fase es un importante insumo que permitirá determinar los principales ... de acceso y limitar el rango de búsqueda de vulnerabilidades dependiendo de los servicios ...ción. Existen una gran cantidad de herramientas que ayudan en la automatización de este ...actividades y lo mejor de todo, es que muchas de ellas se pueden automatizar desde *scripts*

2.1 Reconocimiento y enumeración con Scapy

...es una librería en *Python* que se caracteriza por ser muy potente y flexible a la hora de ... analizar, manipular e inyectar paquetes en un segmento de red. Soporta una gran cantidad ... de red y se utiliza en un gran número de proyectos de seguridad informática en ...se requiere manipular paquetes de red. Además de permitir la manipulación de paquetes, ...es una excelente herramienta para realizar actividades de reconocimiento y escaneo de ...con la ventaja evidente de que permite controlar todo desde un *script* en *Python*. Por otro ...independientemente de la herramienta utilizada, se han ido desarrollando varias técnicas que ...realizar escaneos de puertos y detectar servicios en ejecución, cerrados y filtrados. Cada ...características muy concretas y para un atacante o *pentester* es importante conocerlas bien ... en qué momento resulta conveniente utilizarlas.

...se asumen unos conocimientos básicos sobre el funcionamiento de protocolos tales como ... *UDP*.

2.1.1 TCP Connect Scan

...se realiza una conexión *TCP* a un puerto determinado, se intercambian tres paquetes con las ... *SYN*, *SYN+ACK* y *ACK*. Dichos paquetes representan la comunicación completa, donde tanto

el cliente como el servidor reconocen que se encuentran en disposición de intercambiar paquetes de datos. Se trata evidentemente del mecanismo más fiable para detectar si un puerto se encuentra abierto, pero también suele ser el mecanismo más auditado y vigilado por sistemas de detección y prevención de intrusiones. En este tipo de escaneo, un puerto se encuentra abierto en el caso de que el servidor responda con un paquete que contenga la *flag ACK* después de enviar un *SYN* y se asume que se encuentra cerrado en el caso de que la respuesta por parte del servidor sea un paquete con la *flag RST*.

Script connect_scan.py

```
#! [illegible]
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR
from scapy.all [illegible]

if len(sys.argv) != 3
    print "usage: Python connect_scan.py <ip address> <list of ports separated by colon>"
    exit(1)

src_port = RandShort()
dst_port=22
dst_ip = sys.argv[1]
[illegible]

ports [illegible]
[illegible]
for [illegible]
        [illegible]port),flags="[illegible]")
        [illegible]
            [illegible]
        [illegible]
            [illegible]
                [illegible]st,dport=int(port),flags="AR"))
                [illegible]
        [illegible]
                [illegible]
```

2.1.2 TCP Stealth Scan

Se trata un tipo de escaneo basado en el *TCP Connect* (*three-way handshake*) con la diferencia de que la conexión contra el objetivo en el puerto especificado no se completa íntegramente, ya que es cortada por el cliente antes de que pueda completarse el *handshake*. Consiste simplemente en comprobar el paquete de respuesta del objetivo ante un paquete con la *flag SYN* habilitada. Si el objetivo responde con un paquete que tiene *flag RST* establecida, se asume que el puerto se encuentra cerrado o filtrado.

En el caso de que el paquete de respuesta contenga las *flags SYN+ACK*, se asume que el puerto se encuentra abierto y dado que al atacante no le interesa completar el *handshake* envía un paquete con la *flag RST* para finalizar la conexión.

[illegible]ript stealth_scan.py

```
[illegible]r/python
[illegible]
[illegible] logger("scapy.runtime").setLevel(logging.ERROR)
[illegible] scapy.all import *

[illegible]
[illegible] Python stealth_scan.py <ip address> <list of ports separated by
[illegible]

[illegible]

[illegible].strip().split(':')

[illegible] in scanPorts:
[illegible]

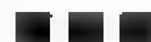
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
        print port+": Open"
    [illegible]ponse.getlayer(TCP).flags == 0x14):
        print port+": Closed"
    [illegible] haslayer(ICMP)):
        [illegible] response.getlay-
[illegible].code) in [1,2,3,9,10,13]):
        print port+": Filtered"
```

[illegible] Escaneo UDP

[illegible] protocolo que se caracteriza por garantizar que las tramas de los paquetes de datos se [illegible] completa y ordenadamente sobre un canal de comunicación entre cliente y servidor [illegible]ado. *UDP* es un protocolo que se caracteriza por enviar lo más rápidamente posible los [illegible] de datos a un destino concreto.

[illegible]colo no garantiza que los paquetes enviados lleguen en orden o que las tramas lleguen [illegible] esto es debido a que en el protocolo *UDP* se asume que el destinatario se encuentra [illegible] para recibir paquetes de datos y en ese modelo no hay cabida para el reconocimiento de la [illegible]ción, como sí ocurre con el *handshake* que se debe establecer entre cliente y servidor *TCP*

[illegible] de servicios que se ejecutan bajo protocolo *UDP*, es un proceso simple y muy rápido, ya [illegible] lleva a cabo el reconocimiento de la comunicación entre ambas partes y para probar si un [illegible] se encuentra abierto, solamente es necesario enviar un paquete *UDP* al puerto en cuestión. En



el caso de que el puerto se encuentre abierto, la respuesta será un paquete UDP, en el caso contrario la respuesta será un paquete ICMP del tipo 3 (Puerto Inalcanzable).

Script udp_scan.py

```
#! /usr/bin/python
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *

[illegible]n(sys.argv) != 3:
    [illegible]
    [illegible]
    exit()

src_port = RandShort()
dst_ip = sys.argv[1]
ports = sys.argv[2]

ports.replace(" ", "")
scanPorts = ports.strip().s[illegible]

for port in scanPorts:
        [illegible](dst=dst_ip)/UDP(dport=int(port)))
        [illegible]sponse))=="<type 'NoneType'>":
            [illegible]
            [illegible] range(0,3)
                [illegible]ppend(sr1(IP(dst=dst_ip)/
UDP(dport=int(port))))
            [illegible]
                [illegible]
                    [illegible]": Open"
            [illegible]
        [illegible]
type)==3 and int(response.getlaye[illegible])
            [illegible]"Closed"
        [illegible]P).type)==3
and int(response.getlayer(ICMP).code) in [illegible]
            print port+": Filtered"
```

2.1.4 ACK Scan

El escaneo *TCP ACK* en realidad no es una técnica para detectar la disponibilidad de un servicio, sino que le permite a un atacante perfilar el objetivo y determinar si hay algún *Firewall* en ejecución en la máquina remota.

Implementar este tipo de escaneo es sencillo, solamente es necesario crear un paquete con la *flag ACK* activada y enviarlo al objetivo, en el caso de que la máquina remota responda con un paquete que tenga la *flag RST* habilitada, se asume que el puerto no se encuentra filtrado por ningún *firewall*. En el caso de que el objetivo no responda o que responda con un paquete *ICMP* del tipo 3, se asume que hay un *Firewall* filtrando los paquetes entrantes en el puerto especificado.

Script ack_scan.py

```
nt.me").setLevel(logging.ERROR)
scapy.all import *

len(sys.argv) != 3:
  print "usage: Python ack_scan.py <ip address> <  t of ports separated by co-

  exit()

  port = RandShort()
  ip = sys.argv[1]
   = sys.argv[2]

    s = ports.strip().split(',')

  port in scanPorts:

      if (str(type(response))=="<type 'NoneType'>"):

                  et.ayer(TCP).flags == 0x4):
                         print "No firewall"
      elif(response.haslayer(ICMP)):
                  if(int(response.getlayer(ICMP).type)==3 and int(response.
  r(ICMP).code) in [1,2,3,9,10,13]):
                         print "Stateful firewall"
```

2.1.5 XMAS Scan

...scaneo se basa en el establecimiento de las *flags PSH, FIN* y *URG*. Al ver los paquetes de ... parecen un árbol de navidad, con varias *flags* establecidas y varias sin establecer, ese es el ... su nombre. Si el puerto en cuestión se encuentra abierto, no habrá respuesta por parte ... en el caso de que retorne un paquete con la *flag RST* indica que el puerto se encuentra ...mente, si devuelve un paquete *ICMP* del tipo 3, el puerto se encuentra filtrado.

... xmas.py

```
  y.all import *
  logging
  .getLogger("scapy.runtime").setLevel(logging.ERROR)

                                 ", "")
```

```
scanPorts = ports.strip().split(':')

for port in scanPorts:
    if port.isdigit():
        response = sr1(IP(dst=destination)/TCP(dport=int(port),flags="FPU"))
        if response is None:
            print "[*] Puerto %s  Open|Filtered " %(port)
        elif(response.haslayer(TCP) and response.getlayer(TCP).flags == 0x14):
            print "[-] Puerto %s Closed" %(port)
        elif(response.haslayer(ICMP)) and int(response.getlayer(ICMP).type)==3:
            print "[-] Puerto %s Filtered" %(port)
    else:
        print "[-] Puerto %s invalido..." %(port)
```

2.1.6 FIN Scan

Un escaneo *FIN*, envía un paquete *TCP* al objetivo con la *flag FIN* establecida. En el caso de q
no exista una respuesta por parte de la máquina remota, se asume que el puerto se encuentra abie
pero si la respuesta es un paquete con la *flag RST* establecida, se asume que el puerto se encue
cerrado.

Script fin_scan.py

```
#! /usr/bin/python
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *
if len(sys.argv) != 3:
    print "usage: Python fin_scan.py <ip address> <list of ports separated by co-
lon>"
    exit()
src_port = RandShort()
dst_ip = sys.argv[1]
ports = sys.argv[2]
ports.replace(" ", "")
scanPorts = ports.strip().split(':')

for port in scanPorts:
        response = sr1(IP(dst=dst_ip)/TCP(dport=int(port),flags="F"))
        if (str(type(response))=="<type 'NoneType'>"):
                print "Open|Filtered"
        elif(response.haslayer(TCP)):
                if(response.getlayer(TCP).flags == 0x14):
                        print "Closed"
        elif(response.haslayer(ICMP)):
                if(int(response.getlayer(ICMP).type)==3 and int(response.
getlayer(ICMP).code) in [1,2,3,9,10,13]):
                        print "Filtered"
```

2.1.7 NULL Scan

Un escaneo *NULL*, envía un paquete *TCP* al objetivo sin ninguna *flag* establecida. En el caso de que la máquina remota no emita ninguna respuesta, se asume que el puerto se encuentra abierto, si la máquina remota devuelve un paquete con la *flag RST*, se asume que el puerto se encuentra cerrado.

Script null_scan.py

```
#! /usr/bin/python
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *
if len(sys.argv) != 3:
    print "usage: Python null_scan.py <ip address> <list of ports separated by co-
lon>"
    exit()
src_port = RandShort()
dst_ip = sys.argv[1]
ports = sys.argv[2]
ports.replace(" ", "")
scanPorts = ports.strip().split(':')
for port in scanPorts:
        response = sr1(IP(dst=dst_ip)/TCP(dport=int(port),flags=""))
        if (str(type(response))=="<type 'NoneType'>"):
                print "Open|Filtered"
            elif(response.haslayer(TCP)):
                    if(response.getlayer(TCP).flags == 0x14):
                            print "Closed"
            elif(response.haslayer(ICMP)):
                    if(int(response.getlayer(ICMP).type)==3 and
int(response.getlayer(ICMP).code) in [1,2,3,9,10,13]):
                            print "Filtered"
```

2.1.8 Window Scan

Este tipo de escaneo funciona igual que el escaneo del tipo *ACK*, con la diferencia de que en este caso, si que se intenta determinar el estado de un puerto verificando el tamaño del campo *window* del paquete devuelto por el objetivo. La respuesta de la máquina remota será un paquete con la *flag RST* establecida, pero en el caso de que el valor del campo *window* sea positivo, se asume que el puerto se encuentra abierto. Si el valor del campo *window* es cero, se asume que el puerto se encuentra cerrado.

Script window_scan.py

```
#! /usr/bin/python
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *
if len(sys.argv) != 3:
    print "usage: Python window_scan.py <ip address> <list of ports separated by
colon>"
    exit()
```

```
src_port = RandShort()
dst_ip = sys.argv[1]
ports = sys.argv[2]
ports.replace(" ", "")
scanPorts = ports.strip().split(':')
for port in scanPorts:
        response = sr1(IP(dst=dst_ip)/TCP(dport=int(port),flags="A"))
                if (str(type(response))=="<type 'NoneType'>"):
                        print "Sin respuesta"
                elif(response.haslayer(TCP)):
                        if(response.getlayer(TCP).window == 0):
                                print "Closed"
                        elif(response.getlayer(TCP).window > 0):
                                print "Open"
```

Las técnicas de escaneo que se han implementado en esta sección con *Scapy*, también se encuentran implementadas en muchas otras herramientas, como por ejemplo, *Nmap*.

2.2 Utilizando NMAP desde Python

NMAP es una potente y robusta herramienta que implementa diferentes técnicas de escaneo de puertos en ordenadores. Su principal objetivo es el de encontrar información detallada sobre un sistema concreto o un segmento de red completo. Incluye un avanzado sistema de detección de servicios, puertos abiertos y otras características bastante interesantes que pueden servir de apoyo a un atacante para descubrir sistemas vulnerables y planificar su ataque. Dentro de las características más interesantes, se incluyen diferentes tipos de técnicas de evasión de sistemas de seguridad como *IDS* o *Firewalls*, así como también un potente motor de *scripting* que permite recolectar información y detectar vulnerabilidades concretas sobre el sistema escaneado. En este apartado, se explicarán las características más interesantes que se incluyen en *NMAP*.

2.2.1 Script Engine

Una de las características más interesantes y potentes que tiene *Nmap* es su motor para crear y ejecutar *scripts* siguiendo la especificación *NSE* (*Nmap Scripting Engine*). De esta forma no solamente es posible encontrar puertos abiertos, sino que también es posible ejecutar rutinas más complejas que permitan filtrar información sobre un objetivo. Actualmente hay una gran cantidad de *scripts* incluidos en *Nmap* para múltiples propósitos, como por ejemplo detectar los métodos *HTTP* soportados por un servidor *web*, si un servicio *SNMP* tiene habilitado algún nombre de comunidad frecuente como *public* o *private*, detectar si el objetivo ejecuta un *WAF* (*Web Application Firewall*), entre muchas otras rutinas que resultan muy útiles para un atacante.

Para ejecutar un *script*, es necesario indicar la opción *--script* junto con el nombre del *script NSE* que se desea ejecutar. En el caso de que el *script* requiera algún argumento, también es necesario incluir la opción *--script-args*. Por otro lado, la opción *-sC* también puede ser utilizada para ejecutar

todos los *scripts* que coincidan con un puerto abierto o servicio en ejecución encontrado durante el proceso de escaneo.

Los *scripts* disponibles en una instalación de *Nmap*, se encuentran disponibles en el directorio "*<NMAP>/scripts*". Si por ejemplo, *Nmap* se ha instalado en un sistema basado en *Debian* con la utilidad *apt-get*, dichos *scripts* probablemente están ubicados en el directorio "*/usr/local/share/nmap/scripts/*".

Uso del script snmp-interfaces

```
nmap -sUV -p161 192.168.1.2 --script="snmp-interfaces" --script-
args="snmpcommunity=public"
Starting Nmap 6.25 ( http://nmap.org ) at 2014-03-17 22:36 CET
Nmap scan report for 192.168.1.2
Host is up (0.00046s latency).
PORT    STATE SERVICE VERSION
161/udp open  snmp    net-snmp
| snmp-interfaces:
|   lo
|     IP address: 127.0.0.1  Netmask: 255.0.0.0
|     Type: softwareLoopback  Speed: 10 Mbps
|     Status: up
|     Traffic stats: 17.16 Kb sent, 17.16 Kb received
|   Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
|     IP address: 192.168.1.10  Netmask: 255.255.255.0
|     MAC address: 08:00:27:39:73:17 (Cadmus Computer Systems)
|     Type: ethernetCsmacd  Speed: 100 Mbps
|     Status: up
|_    Traffic stats: 1.42 Kb sent, 12.91 Kb received
MAC Address: 08:00:27:39:73:17 (Cadmus Computer Systems)
Service detection performed. Please report any incorrect results at http://nmap.
org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 5.97 seconds
```

Uso del script snmp-netstat

```
nmap -sUV -p161 192.168.1.2 --script="snmp-netstat" --script-
args="snmpcommunity=public"
Starting Nmap 6.25 ( http://nmap.org ) at 2014-03-17 22:38 CET
Nmap scan report for 192.168.1.2
Host is up (0.00046s latency).
PORT    STATE SERVICE VERSION
161/udp open  snmp    net-snmp
| snmp-netstat:
|   TCP  0.0.0.0:22           0.0.0.0:0
|   TCP  0.0.0.0:80           0.0.0.0:0
|   TCP  0.0.0.0:443          0.0.0.0:0
|   TCP  127.0.0.1:199        0.0.0.0:0
|   TCP  127.0.0.1:199        127.0.0.1:35822
|   TCP  127.0.0.1:199        127.0.0.1:35825
|   TCP  127.0.0.1:199        127.0.0.1:35826
|   TCP  127.0.0.1:35822      127.0.0.1:199
|   TCP  127.0.0.1:35825      127.0.0.1:199
```